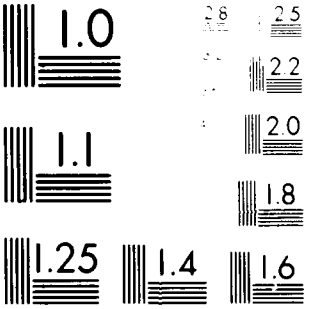
AD-A112 429 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 15/4
A PROTOTYPE PROGRAM FOR TARGET INFORMATION.(U)
JUN 81 R J COULTER
UNCLASSIFIED NPS52-81-007 NL

1 OF 2
AD A
112429


1.0
2.8
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

PHOTOGRAPH THIS SHEET

AD-A112429

DTIC ACCESSION NUMBER

II

LEVEL

I

INVENTORY

Naval Postgraduate School
Monterey, CA

A Prototype Program for Target Information

DOCUMENT IDENTIFICATION

Jun. 81

Coulter, Ronald J.

Rept. No. NPS52-81-007

DISTRIBUTION STATEMENT A
Approved for public release,
Distribution Unlimited

DISTRIBUTION STATEMENT

ACCESSION FOR		
NTIS	GRA&I	☑
DTIC	TAB	☐
UNANNOUNCED		☐
JUSTIFICATION		
BY		
DISTRIBUTION /		
AVAILABILITY CODES		
DIST	AVAIL AND/OR SPECIAL	
A		

DISTRIBUTION STAMP

DTIC
ELECTE
MAR 23 1982
S D
D

DATE ACCESSIONED

DTIC
COPY
INSPECTED

8

DATE RECEIVED IN DTIC

PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-DDA-2

DTIC FORM OCT 79 70A

DOCUMENT PROCESSING SHEET

NPS52-81-007

NAVAL POSTGRADUATE SCHOOL

Monterey, California

A PROTOTYPE PROGRAM FOR TARGET INFORMATION

Ronald J. Coulter

June 1981

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral J. J. Ekelund
Superintendent

D. A. Schrady
Acting Provost

The work reported herein was supported by the Microcomputer Laboratory, Department of Computer Science, Naval Postgraduate School, Monterey, California.

This report was prepared by:

RONALD J. COULTER
LTCOL, U.S. Marine Corps

Reviewed by:

GORDON H. BRADLEY, Chairman
Department of Computer Science

Released by:

WILLIAM M. TOLLES
Dean of Research

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NPS52-81-007	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Prototype Program for Target Information		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Ronald J. Coulter LTCOL USMC		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93940		12. REPORT DATE June 1981 13. NUMBER OF PAGES 128
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Microcomputer, User interface, Target Information, Data Base, FSCC (Fire Support Coordination Center), Fire Support Coordination, Marine Corps, UCSD Pascal, Amphibious Operations

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis presents the specification, design and implementation of a prototype microcomputer system for the target information section of the Marine Corps fire support coordination center. Currently, the target information section uses a series of index cards, handwritten lists, acetate covered battle maps and grease pencils to perform the target information functions.

The thesis examines and analyzes these functions in detail and proposes

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20.

a solution in the form of a system, data base and interactive user design. The resultant Microcomputer System for Traget Information (MISTI) employs an ALTOS Z-80 microcomputer, the UCSD Pascal operating system, a user friendly interface and data base technology. It is proposed as an interim system until the Marine Integrated Fire and Air Support System (MIFASS) becomes operational.

TABLE OF CONTENTS

A PROTOTYPE PROGRAM FOR TARGET INFORMATION

Introduction

More and more of the applications of modern amphibious warfare have turned to computerized solutions, from real-time combat systems to the data bases that control the men, materiel and resources needed to wage war. The products of the technological explosion have enabled the Navy-Marine Corps amphibious team to do more, to do it faster and to do it with a degree of efficiency and accuracy previously unobtainable.

This evolution of modern technology has not yet reached the Marine Corps tactical command posts established on the beachhead. The target information section of the landing force fire support coordination center (FSCC) plays a significant role in the conduct of effective coordination of tactical air, artillery and naval gunfire support on targets of high priority. Yet the target information officer and his staff accomplish their important task by the use of index card files, cross-reference files, hand written lists of targets and colored grease pencils on acetate-covered tactical maps. This method is time consuming, slow in response to inquires about target information, tedious and

difficult to maintain in a current status and does not provide information in a sufficiently timely and accurate manner. It is 40 year old technology in the age of computers.

The requirement to automate many of the functions of the tactical command post has been identified and the command post of the future is being planned for and developed now. Until it arrives, there is a need to provide an interim capability to the landing force. An automated solution to the target information function will simplify the task of the target information section considerably, will provide rapid, accurate and timely target information to the members of the FSCC, and can be made operational now, five full years before the planned introduction of the computerized command post.

This report contains a prototype program for target information which will improve the operational capability of the landing force FSCC and show that the implementation of a suitable and effective target information system is possible. This implementation and design of a working prototype will increase operational effectiveness immediately as well as provide a testbed and learning model for the future automated command post. The prototype is designed to perform all the duties and functions of the target information section as currently stated in doctrinal publications. The interim system will hopefully contribute

to the development of the future system and identify areas of concern and improvement before the future Marine Corps system becomes operational.

Background

An important aspect of amphibious fire support coordination (the planning and execution of tactical air, artillery and naval gunfire support so that targets are adequately covered by a suitable weapon or group of weapons) is the function of target information. One of the major duties of the fire support coordinator, that member of the landing force staff responsible for coordination of fire support, is to ensure that the fire support coordination center receives and disseminates available target information to all staff sections and commands requiring the information. He also must work closely with the target information officer and the commander and his staff in the selection of targets and assignment of classification and attack priorities.

Target information is the direct application of combat intelligence to fire support and is a key to the proper employment of supporting arms in conjunction with each of the plans of the amphibious operation. Effective fire support coordination and the planning of amphibious operations generate a continuing requirement for target acquisition, dissemination, evaluation and recommendation

for attack.

To accomplish this important task, the commander of the amphibious task force assigns a target intelligence officer to the supporting arms coordination center (SACC). This officer operates the target information center (TIC) and works closely with the air intelligence officer, the landing force targeting representatives and the supporting arms coordinator. The commander of the landing force has a target information officer (TIO) who operates the target information section (TIS) as an integral part of the landing force fire support coordination center and a target intelligence officer who functions in the landing force intelligence center.

The Navy staff uses a computerized target information system which is part of the shipboard Amphibious Support Information System (ASIS) and maintains the list of targets as part of a data base. Target information operations in the SACC are thus computerized and, while the ASIS target system is not the most modern of data base systems, it is efficient, effective and fast. When the functional responsibility for maintaining targets is passed ashore to the landing force TIO, the computer system is replaced by an index card filing system, which , while effective, is neither fast nor efficient by comparison. Additionally, the index card system lends itself to inaccuracies and omissions in target data, particularly when the information must be

maintained in a timely manner. The tactical requirement for accurate and timely target information is no less critical or important when the landing force is on the beach, yet the system to accomplish this task is antiquated and cumbersome.

The staff of the TIS manually transfers the target information data contained in the ASIS data base to 5 by 8 inch target cards. After duplicating the entire target file, the TIS must construct a cross reference file to list the target by grid location and a cross-index file to keep track of certain types of targets. In addition to the target cards, the TIS also makes up lists of particular categories of targets which may be of interest or value to members of the FSCC.

The TIS obtains intelligence information from landing force and supporting arms agencies, converts this to target information and enters the information into the target card files. The information is made available to the supporting arms representatives in the FSCC and, based on the TIO's recommendations, a decision is made when and how to attack a particular target. Results of attacks on targets, front line reports and intelligence information are used to refine the target list and delete or deprioritize those targets that present a diminished threat to the landing force.

Access to specific information from the target list (for example, more than one category of the cross-index files) requires physically searching through each list and

constructing sub-lists to determine the appropriate information. The constant availability of timely and accurate target information is required for the effective employment of supporting arms and planning of fire support. The TIS plays a key role in providing this information and the constant process of adding to the target list, selecting targets for attack and deleting targets once neutralized is performed by the TIS staff using the target card file.

One of the most complex aspects of modern amphibious warfare is the control and coordination of supporting arms particularly in the transition of responsibility from the Navy in amphibious ships to the Marine Corps combat units ashore. The grease pencils, map boards and field radios that have served Marines so well since the days of Guadalcanal will, in the future, be eclipsed by the automated system called the Marine Integrated Fire and Air Support System (MIFASS).

MIFASS is part of the Marine Corps integrated command and control system called MTACCS (Marine Tactical Command and Control Systems), a collection of eight major systems which will give the Marines a capability of exercising real-time command and control of combat forces in the post-1980 time frame. MIFASS is designed to perform the functions of the fire support coordination center, (FSCC) the direct air support center (DASC) and, to a degree, the artillery fire direction center (FDC) at one central

location called the Fire and Air Support Center (FASC).

It is a distributed processing system in which microcomputers control interactive display devices, manage data bases, perform computational tasks and drive printers to provide hard-copy records of messages and operator decisions. It is currently in full scale engineering development with an initial operational capability planned for the 1986-1987 time frame. MIFASS addresses the requirement for target information by providing the TIO with a digital display device which will have both a graphical representation of the target on a battle map and a video screen for alphanumeric display of target information.

Statement of the Problem

An automated solution to the target information function will not be realized until the introduction of the MIFASS computers into the Fleet Marine Forces. Until such time as the system is delivered, the target information function of the FSCC is tied to the current doctrine and the target card filing system.

In this report, an interim solution to the problem of automating the target information function of the FSCC is presented. It computerizes those basic functions of the TIS in a simple, inexpensive and effective manner. It simplifies the tasks of the TIS, provides a mechanism for rapid and accurate retrieval of target information and could improve

the operational capability of the FSCC.

Nature of the Solution

The design task is broken down into three distinct parts, each of which is influenced by the dual constraints of a microcomputer environment and a friendly user interface. The design is specifically addressed in the thesis which is supported by this program listing report.

The design of the physical and logical data base is influenced by the desire to have a simple yet sufficiently informative data model, a rapid, real-time response and a restricted, single application system. The system design is influenced by the microcomputer environment which restricts the user both in main memory space and the speed of access to secondary storage and the requirement for an effective interactive system for a non-sophisticated user.

The design of the software to implement both the data base and the system is overwhelmingly influenced by the requirement that the system support real-time, interactive processing of a casual, non-programmer. Termed "Marine proof" in the vernacular, it requires a sophisticated interface employing user friendly dialogue techniques to ensure that the operation is simple and efficient. For this reason, and to facilitate system portability, a microcomputer compatable high level programming language (UCSD Pascal) is used.

The report is divided into two sections. The first is the source code listing, by module, of the Microcomputer System for Target Information (MISTI) program. The second is a listing of the text files used in the interactive user interface which complements much of the prototype program. The reader is referred to the Naval Postgraduate School masters thesis [1] for additional details on the specifications, design and implementation of the system. This report is issued in conjunction with and as a key element of the thesis.

---

1. Coulter, R.J., A Microcomputer System for Target Information in the Fire Support Coordination Center: A Data Base Approach, Masters Thesis, June 1981.

Senan, A., and Sinombing, T. M., Database Management System for Microcomputers, Masters Thesis, Naval Postgraduate School, 1979.

Shneiderman, B., Software Psychology: Human Factors in Computer and Information Systems, Winthrop Publications Inc., 1980.

Smith, L. B., The Use of Interactive Graphics to Solve Numerical Problems, Communications of the ACM, 1970.

Snoderass, R., A Sophisticated Microcomputer User Interface, Procedings of the Third Symposium on Small Systems, 1982.

APPENDIX--A

The source code listing for the MISTI system is organized by functional module. The list is not a compiled listing but is separate by function and by module. In an effort to decrease user confusion, the following outline shows the logical organization of the program. The segment procedures are marked by a *.

```
                          .....GLOBALS
                          :
                          :....QUERY*          ....EDA*
                          :                    :
                          :....TARGET*........:...TGTPROCS
                          :                    :
INTERFACE.............:                    :
                          :                    :...ADDTARGET
                          :....INIT*           :
                          :                    :...CHANGE
                          :....INFORM*         :
                          :                    :...TARGET
                          :
                          :....UTILITY*
```

The Interface module contains include statements which instruct the compiler to compile the program in the proper logical order. The segment procedures are the first procedures to be compiled. This necessitates the declaring of many of the system routines in the beginning of the listing. The UCSD Pascal include statement allows the user to identify the volume name as well as the file name. "Store" is the name of the volume which contains the source code, thus, it appears in the include commands.

```
(*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*)

(*          MAIN PROGRAM.....INTERFACE          *)

program interface (input,output);


        (*$Istore:globals.text*)


procedure clear; forward;          procedure delay; forward;
procedure prompt; forward;         procedure spacebar; forward;
procedure select; forward;         procedure returnbar; forward;
procedure menuerror; forward;      procedure halt; forward;
procedure error1; forward;         procedure error2; forward;
procedure lines(y : integer); forward;
procedure loadmsg; forward;
procedure getfile( filename : string); forward;


        (*$Istore:query.text*)
        (*$Istore:bda.text*)
        (*$Istore:tgtprocs.text*)
        (*$Istore:addtarget.text*)
        (*$Istore:change.text*)
        (*$Istore:target.text*)
        (*$Istore:init.text*)
        (*$Istore:inform.text*)
        (*$Istore:utility.text*)


           procedure clear;

               begin
                 write(chr( 61 ));
               end;


           procedure prompt;

                begin
                  write(prompter);
                end;

           procedure spacebar;

                begin
                  writeln(spacebr);
                  prompt;
                  repeat
                  read(ch);
                  until ch = ' ';
                end;
```

```
procedure select;

      begin
        writeln(selectop);
        prompt;
      end;


procedure lines;

      var   yy : integer;

      begin
        for yy := 1 to y do
        writeln;
      end;


procedure returnbar;

      begin
        writeln(returner);
        prompt;
        repeat
          read(cn);
        until eoln(input);
      end;


procedure loadmsg;

begin
  lines(3);
  write(chr(14));
  writeln('                PROGRAM BEING READ IN.....PLEASE WAIT');
  write(chr(24));
  writeln(chr(29));
end;


   procedure delay;

      var
        x, y, z : integer;

      begin
        z := 0;
        x := 0 ;
        repeat
          for y := 1 to 100 do
            z := z + 1 ;
          x := x + 1 ;
```

```
      until x = 10 ;
    end;


procedure halt;

    var
       z : integer;

    begin
      lines(2);
      write('      System halting...');
      for z := 1 to 10 do
      begin
        delay;
        write(dot);
      end;
    end;


procedure error1;

begin
  lines(1);
  writeln('          The proper format is a number from the');
  writeln('          options listed above. Please press the');
  writeln('          RETURN key and reenter your choice.');
  lines(1);
  prompt;
end;


procedure menuerror;

begin
  lines(1);
  if eoln (input) then
  begin
    error1;
    readln;
  end
  else error1;
  readln;
end;


procedure error2;


begin
  lines(3);
```

```
    writeln('        The target identifier does not currently');
    writeln('        exist in the target file. Please reenter');
    writeln('        the target identifier.');
    lines(1);
    numbout := numbout + 1;
    if numbout = 3 then
    begin
      lines(1);
      writeln('        To leave this procedure, type a C');
      writeln('        followed by pressing the RETURN key.');
      lines(1);
      numbout := 0 ;
    end;
  end;


    procedure getfile;

    (*  filename : string  declared forward   *)

    var   buffer : string;
          usermessage : text;

    begin
      reset(usermessage,filename);
      repeat
        begin
          readln(usermessage,buffer);
          writeln(buffer);
        end;
      until eof(usermessage);
      close(usermessage,lock);
    end;


(*.............................................*)

    procedure buildempty;

    var  j : integer;

  begin
    with emptyTrec do
      begin
        tnum := '000000';
        alt := '    ';
        loc := '00000000';
        acc := ' ';
        class := ' ';
        pri := ' ';
        ttype := ' ';
```

```
      sa := ' ';
      stat := ' ';
      desc :=                                                  ';
      rem := desc;
      maprefer :=                          ';
      sour := maprefer;
      photonum :=                    ';
      DTGact :=          ';
      photocord := loc;
      for i := 1 to numbervar do
        flag[i] := off;
      for i := 1 to 5 do
      begin
        BDA[i].DTGsurv := DTGact;
        BDA[i].fireunit :=          ';
        BDA[i].ntrnds          ';
        BDA[i].damrep :=  ';
        BDA[i].damass :=  ';
        BDA[i].BDAtext := desc;
      end;
    end;
  with emptyCrec do
    begin
      tnum :=          ';
      alt :=      ';
      loc :=          ';
      class :=  ';
      pri :=  ';
      acc :=  ';
      ttype :=  ';
      sa :=  ';
      desc :=                          ';
    end;
  end;



          procedure buildmap;

var   j : integer;

begin
  i := 0;
  j := 0;
  while not eof(target) do
    begin
      seek(target,i);
      get(target);
      tgtmap[i] := target^.tgtrec.tnum;
      i := i + 1;
      j := j + 1;
      if j = 10 then
```

```
        begin
          write(dot);
          j := k;
        end;
      end;
    end;


        procedure openfiles;

var   io : integer;

begin
  filecheck := false;
  clear;
  lines(6);
  loadmsg;
  lines 4);
  write('      Opening file...');
  write(dot);
  (*$I-*)
  reset (target,'#5:targetfile.data');
  (*$I+*)
  io := ioresult;
  if io in [4,5,9] then
  begin
    clear;
    writeln(chr(7));
    lines(8);
    writeln(chr(14));
    writeln('            ***   NO DISK IN DRIVE  B  ***');
    write(chr(24));
    writeln(chr(29));
    lines(5);
    writeln('  Insert TARGET diskette in drive B and restart system');
    halt;
    getout := true;
    exit(openfiles);
  end;
  if io <> 0 then
  begin
    filecheck := true;
    initialize;
  end;
  write(dot);
  (*$I-*)
  reset (QT,'#5:queryfile.data');
  (*$I+*)
  if not filecheck then
  begin
    io := ioresult;
    if io <> 0 then
     begin
```

```
      filecheck := true;
      initialize;
    end;
  end;
  write(dot);
  buildmap;
  write(dot);
  filecheck := false;
end;


    procedure password;
       var
          n,z : integer;
          user : string;
          valid : boolean;


       begin {1}
         valid := false;
         clear;
         n := 1;
         lines(5);
         while (not valid) and (n <= 5) do
         begin {2}
           writeln('    PLEASE ENTER PASSWORD AND PRESS RETURN KEY');
           prompt;
           readln(keyboard,user);
           z := 1;
           while (not valid) and (z <= 5) do
           if userid[z] = user then
           begin{3}
             passwrd := true;
             valid := true
           end{3}
           else z := z + 1 ;
           if not valid then
             begin{4}
               n:= n + 1 ;
               lines(3);
               writeln('     ** PASSWORD INCORRECT **');
               writeln(chr(7));
               lines(1);
               if n > 5 then
               begin{5}
       writeln('        Please refer to the password instructions');
       writeln('        in the target information system handbook');
       writeln('        for the proper input.');
                 lines(4);
                 halt;
                 lines(4);
       writeln('       **  To restart system type R  **');
                 exit(password);
               end;{5}
```

```
            end;{4}
          end;{2}
        end;{1}



      procedure welcome1; forward;

      procedure welcome;

 begin
   clear;
   writeln(dots);
   writeln('        WELCOME TO THE TARGET INFORMATION SYSTEM');
   writeln(dots);
   lines(1);
   writeln('This program is a prototype target information system for');
   writeln('the Fire Support Coordination Center. It is designed to be');
   writeln('used by the personnel of the target information section of');
   writeln('the landing force FSCC in accordance with the principles');
   writeln('outlined in FMFM 7-1 (Fire Support Coordination).');
   lines(2);
   writeln('WARNING:');
   write(chr(14));
   writeln('      *** THIS FILE CONTAINS CONFIDENTIAL MATERIAL ***');
   write(chr(24));
   writeln(chr(29));
   welcome1;
end;


      procedure welcome1;

    begin
writeln(' The diskette file contains targets which are normally classified');
writeln('confidential. The diskettte and all the backup copies should be ');
writeln('handled as normal confidential documents and properly safeguarded.');
writeln('Targets of a higher classification should not be entered on this');
writeln('file. Current emergency destruction procedures for confidential');
writeln('material apply. Re-initializing the system removes all classified');
writeln('information.');
     lines(1);
     spacebar;
   end;



      procedure welcome2;

    begin
      clear;
      lines(5);
```

```
writeln('    If at any time you become confused, in doubt about what');
writeln('to do next or what values to enter when you receive a prompt');
writeln('from the system ( ==> ), you can receive help or information');
writeln('by typing a ?.');
lines(4);
writeln('    If you need more information on how to operate the system,');
writeln('doctrinal guidelines for target information, security requirements');
writeln('or the types of formats used for target information, select option');
writeln('number 1 from the main command menu which follows.');
      lines(2);
      spacebar;
    end;



            procedure mminfo;

          begin
            clear;
            writeln(stars);
            lines(1);
            getfile('mminfo1.text');
            lines(1);
            spacebar;
            clear;
            lines(2);
            getfile('mminfo2.text');
            lines(1);
            spacebar;
            clear;
            lines(5);
            getfile('mminfo3.text');
            lines(5);
            spacebar;
          end;


            procedure mainmenu;

    begin
      clear;
      writeln(stars);
      writeln('     Target Information System Main Command Menu');
      writeln(dots);
      lines(1);
      writeln('    The options are:');
      lines(1);
      writeln('          1.  System Information');
      writeln('          2.  Work on Target File');
      writeln('          3.  Create a Special Target List');
      writeln('          4.  Perform Utility Functions');
      writeln('          5.  Initialize a New System');
      writeln('          6.  Information about this Menu');
      writeln('          7.  Halt Operation');
```

```
  lines(2);
end;



begin (* interface*)

   getout := false;
   restart := false;
   menuloop := false;
   numbout := 0;
   passwrd := false;
   userid[1] := 'COULTER';
   userid[2] := 'E';
   userid[3] := 'e';
   userid[4] := 'MARINE';
   userid[5] := 'marine';
   menuchar := ['1','2','3','4','5','6','7','7'];
   password;
   if not passwrd then exit(interface);
   buildempty;
   openfiles;
   if getout then exit(interface);
   welcome;
   clear;
   welcome2;
   repeat
     mainmenu;
     select;
     read(ch);
     if ch in menuchar then
       begin
       if eoln (input) then readln;
       case ch of
           '1' : begin
                   loadmsg;
                   inform;
                 end;
           '2' : begin
                   loadmsg;
                   targetmod;
                 end;
           '3' : begin
                   loadmsg;
                   query;
                 end;
           '4' : begin
                   loadmsg;
                   utility;
                 end;
           '5' : begin
                   loadmsg;
```

```
                    initialize;
                    if restart then
                      begin
                        write('       Processing....');
                        buildmap;
                        lines(2);
                        writeln('                    FUNCTION  COMPLETE');
                        lines(1);
                        spacebar;
                      end;
                  end;
            '6','?' : mainfo;
            '7' : begin
                    getout := true;
                    halt;
                    clear;
                  end;
          end
        end
      else menuerror;
      until getout = true;
    end.


(*+++++++++++++++++ END OF INTERFACE +++++++++++++++++++++*)
```

```
        (*    GLOBALS.TEXT    *)

const

dot = '.';
stars = '*****************************************************************';
dots = '.................................................................';
spacebr = '        PLEASE PRESS SPACEBAR TO CONTINUE';
returner = '      PLEASE PRESS RETURN TO CONTINUE';
prompter = '      ==>';
selectop = '      PLEASE ENTER OPTION NUMBER ';
return = '  Return to Previous Menu';
numbervar = 19;
numbertgt = 300;


 type
     mvalue = set of char;
     state = (on,off);

      battledam = packed record
         DTGsurv : string[7];
         fireunit : string[8];
         ntrnds : string[10];
         damrep : char;
         damass : char;
         BDAtext : string[40]
      end;

      terrec = packed record
           flag : packed array [1..numbervar] of state;
           tnum : string[6];
           loc : string[8];
           alt : string[4];
           desc : string[40];
           class : char;
           pri : char;
           stat : char;
           ttype : char;
           sa : char;
           rem : string[40];
           naprefer : string[20];
           sour : string[20];
           photonum : string[15];
           DTGact : string[7];
           photocord : string[8];
           acc : char;
           BDA : packed array [1..3] of battledam
         end;

          querytgt = packed record

                                  24
```

```
            flag : state;
            ttype : char;
            class : char;
            sa : char;
            pri : char;
            acc : char;
            stat : char;
            tnum : string[8];
            loc : string[6];
            alt : string[4];
            desc : string[20]
          end;

      targetmap = packed array [0..numtertgt] of string[6];
      gridloctap = packed array [0..numbertgt] of string[6];
      Qtarget = packed array [0..numbertgt] of querytgt;


var
      gridmap : gridlocmap;
      tatmap : targetmap;
      target : file of record tgtrec : tarrec end;
      QT : file of record querrec : querytgt end;
      nochar, ch : char;
      restart, menuloop, passwrd, getout : boolean;
      userid : packed array [1 .. 5] of string;
      menuchar, menucar : nvalue;
      nostring, buffer, str : string;
      recnum, numbout, numbcheck, tlp, range, flag : integer;
      filecheck, current, mandatoryitem, helpme, finished : boolean;
      endEDA, ok, trap, done, quit : boolean;
      x, i, ii, EDAcounter, n : integer;
      emptytrec, currentgt : tarrec;
      emptyQrec, currentQT : querytgt;
      database : Qtarget;


(*..................... SYSTEM GLOBALS .........................*)
```

```
            (*      QUERY .TEXT      *)

segment procedure query;

      var     charmenu : mvalue;
              tellused, first : boolean;
              amount, left, searcher, count, reccount : integer;
              index : char;
              Scheck, Pcheck, Acheck, statcheck : string[16];
              Tcheck, Ccheck, emptystring : string[10];
              cat : array [1..6] of string;


     procedure getdatabase;

     var  j : integer;

     begin
       clear;
       lines(4);
       writeln('   Data base being loaded into memory.....Please wait');
       lines(5);
       write('            loading...');
       i := 0;
       j := 0;
       close(QT,lock);
       reset(QT,'#5:queryfile.data');
       while not eof(QT) do
       begin
         seek(QT,i);
         get(QT);
         database[i] := QT^.querrec;
         i := i + 1;
         j := j + 1;
         if j = 10 then
         begin
           write(dot);
           j := 0;
         end;
       end;
       lines(6);
       write('                    LOADING COMPLETE');
       reccount := i;
       for i := 1 to 4 do
         delay;
     end;
```

```
procedure moresearch; forward;

procedure searchdatabase;


begin
  for i := 0 to reccount - 1 do
  begin
  case searcher of
    1 : begin
          if first then
          begin
           if database[i].ttype = index then database[i].flag := on
          end
          else if (database[i].flag = on) and (database[i].ttype <> index)
          then database[i].flag := off;
        end;
    2 : begin
          if first then
          begin
           if database[i].class = index then database[i].flag := on
          end
          else if (database[i].flag = on) and (database[i].class <> index)
          then database[i].flag := off;
        end;
    3 : begin
          if first then
          begin
           if database[i].sa = index then database[i].flag := on
          end
          else if (database[i].flag = on) and  (database[i].sa <> index)
          then database[i].flag := off;
        end;
    4 : begin
          if first then
          begin
           if database[i].pri = index then database[i].flag := on
          end
          else if (database[i].flag = on) and (database[i].pri <> index)
          then database[i].flag := off;
        end;
    5 : begin
          if first then
          begin
           if database[i].acc = index then database[i].flag := on
          end
          else if (database[i].flag = on) and (database[i].acc <> index)
          then database[i].flag := off;
        end;
    6, 7 : moresearch;
    end;
    end;
  amount := 0;
  for i := 0 to reccount - 1 do
```

```
    if database[i].flag = on then amount := amount + 1;
  lines(2);
  writeln('            Number of targets in special list is ',amount);
  lines(2);
  spacebar;
    index := ' ';
    first := false;
  end;


    procedure moresearch;

begin
  case searcher of
    6 : begin {active}
          if first then
          begin
            if (database[i].stat = '1') or (database[i].stat = '2') or
            (database[i].stat = '3') or (database[i].stat = '4') then
            database[i].flag := on
          end
          else if (database[i].flag = on) and ((database[i].stat = '5') or
            (database[i].stat = '6')) then database[i].flag := off;
        end;
    7 : begin {inactive}
          if first then
          begin
            if (database[i].stat = '5') or (database[i].stat = '6') then
            database[i].flag := on
          end
          else if (database[i].flag = on) and ((database[i].stat <> '5') or
            (database[i].stat <> '6')) then database[i].flag := off;
        end;
    end;
  end;


      procedure DBtype;

      begin
        repeat
        clear;
        lines(2);
        getfile('typemenu.text',);
        writeln('            R.',return);
        lines(1);
        select;
        read(cn);
        if cn in charmenu then
        begin
          if eoln(input) then readln;
          if cn in ['Q','q','R','r'] then exit(DBtype)
```

```
    else if ch = '7' then
      begin
        clear;
        getfile('tgttype.text');
        spacebar;
      end
    else begin
           searcher := 1;
           left := left - 1;
           index := ch;
           count := count + 1;
           case index of
             '1' : cat[count] := 'TANK';
             '2' : cat[count] := 'SEAD';
             '3' : cat[count] := 'INST';
             '4' : cat[count] := 'CBAT';
             '5' : cat[count] := ' OP ';
             '6' : cat[count] := 'TERR';
             '7' : cat[count] := 'VEH ';
             '8' : cat[count] := 'FORT';
             '9' : cat[count] := 'MISC';
           end;
           searchdatabase;
           exit(DBtype);
         end
  end
  else menuerror;
  until menuloop = true;
end;


procedure DBclass;

var  temp : string[2];

begin
  temp := '  ';
  repeat
    clear;
    lines(1);
    getfile('classmenu.text');
    writeln('            6.',return);
    lines(2);
    select;
    read(ch);
    if ch in charmenu then
    begin
      if eoln(input) then readln;
      if ch in ['S','s','q','R','r'] then exit(DBclass)
      else if ch = '7' then
        begin
          clear;
          getfile('class.text');
```

```
          spacebar;
        end
      else begin
             searcher := 2;
             left := left - 1;
             count := count + 1;
             case ch of
               '1' : index := 'A';
               '2' : index := 'B';
               '3' : index := 'C';
               '4' : index := 'D';
               '5' : index := 'E';
             end;
             case index of
               'A' : temp := ' A';
               'B' : temp := ' B';
               'C' : temp := ' C';
               'D' : temp := ' D';
               'E' : temp := ' E';
             end;
             cat[count] := concat('Class',temp);
             searchdatabase;
             exit(DBclass);
           end
    end
    else menuerror;
    until menuloop = true;
  end;


procedure DBSAassgn;

begin
  repeat
  clear;
  lines(2);
  getfile('samenu.text');
  writeln('                    R.',return);
  lines(1);
  select;
  read(ch);
  if ch in charmenu then
  begin
    if eoln(input) then readln;
    if ch in ['R','r','e','q'] then exit(DBSAassgn)
    else if ch = '?' then
      begin
        clear;
        getfile('sa.text');
        spacebar;
      end
    else begin
           searcher := 3;
```

```
            left := left - 1;
            index := ch;
            count := count - 1;
            case index of
              '1' : cat[count] := 'ARTY';
              '2' : cat[count] := 'NGF ';
              '3' : cat[count] := 'AIR ';
              '4','5','6','7' : cat[count] := 'COMB';
              '8' : cat[count] := 'OTHR';
              '9' : cat[count] := 'NONE';
            end;
            searchdatabase;
            exit(DPSAassgn);
          end
  end
  else menuerror;
  until menuloop = true;
end;


procedure DBpri;

var temp : string[3];

begin
  repeat
  clear;
  lines(1);
  getfile('tetprimenu.text');
  writeln('                    5.',return);
  lines(1);
  select;
  read(ch);
  if ch in charmenu then
  begin
    if eoln(input) then readln;
    if ch in ['5','R','r','C','q'] then exit(DBpri)
    else if ch = '7' then
      begin
      clear;
      getfile('priority.text');
      spacebar;
    end
    else begin
            searcher := 4;
            index := ch;
            left := left - 1;
            count := count + 1;
            case index of
              '1' : temp := 'I  ';
              '2' : temp := 'II ';
              '3' : temp := 'III';
              '4' : temp := 'IV ';
```

```
            end;
            cat[count] := concat('Pri ',temp);
            searchdatabase;
            exit(DBpri);
          end
    end
    else menuerror;
    until menuloop = true;
  end;


  procedure DBacc;

  begin
    repeat
    clear;
    lines(2);
    getfile('tetaccmenu.text');
    writeln('                    5.',return);
    lines(1);
    select;
    read(ch);
    if ch in charmenu then
    begin
      if eoln(input) then readln;
      if ch in ['B','R','P','A','W'] then exit(DBpri)
      else if ch = 'T' then
        begin
          clear;
          getfile('tetacc.text');
          spacebar;
        end
      else begin
            searcher := 5;
            index := ch;
            left := left - 1;
            count := count + 1;
            case index of
              '1' : cat[count] := 'CONFIRMED';
              '2' : cat[count] := 'PROBABLE  ';
              '3' : cat[count] := 'POSSIBLE  ';
              '4' : cat[count] := 'UNKNOWN   ';
            end;
            searchdatabase;
            exit(DBacc);
          end
    end
    else menuerror;
    until menuloop = true;
  end;
```

```
  procedure Dstatus;

  var  act : string[8];
       loop : boolean;

begin
  act := '        ';
  menuchar := ['1','2','?'];
  loop := false;
  repeat
    clear;
    lines(5);
    writeln('      ENTER TARGET STATUS--ACTIVITY');
    lines(1);
    writeln('       The options are:');
    lines(1);
    writeln('          1.  Active');
    writeln('          2.  Inactive');
    lines(1);
    writeln('        PLEASE ENTER OPTION NUMBER AND PRESS RETURN');
    prompt;
    read(ch);
    if ch in menuchar then
    begin
      if not eoln(input) then readln;
      if ch = '1' then
      begin
        searcher := 6;
        searchdatabase;
        loop := true;
        act := ' ACTIVE ';
      end;
      if ch = '2' then
      begin
        searcher := 7;
        searchdatabase;
        loop := true;
        act := 'INACTIVE';
      end;
      if ch = '?' then
      begin
        lines(1);
writeln('    An Active target is one which is found in the target list or' ;
writeln('the list of targets. An inactive target is in the inactfile.');
        lines(1);
        spacebar;
      end;
      end
      else if (ch in ['Q','q']) or (eoln(input)) then exit(Dstatus)
      else menuerror;
    until loop = true;
  left := left - 1;
```

```
  count := count + 1;
  cat[count] := act;
end;



    procedure catmenu;

  begin
    writeln('              Categories for Special Listing');
    writeln(lots);
    lines(1);
    writeln('  The listing can contain ',left,' items from the below menu:');
    lines(1);
    writeln('              1.  Target type                    ',Tcheck);
    writeln('              2.  Classification                 ',Ccheck);
    writeln('              3.  Supporting air assigned        ',Scheck);
    writeln('              4.  Priority                       ',Pcheck);
    writeln('              5.  Accuracy                       ',Acheck);
    writeln('              6.  Status                         ',statcheck);
    writeln('            * F.  Process information');
    lines(1);
    writeln('        Special list currently contains ',amount,' targets.');
    if amount <= 0 then writeln('                    Please start a new listing.');
    lines(1);
  end;



    procedure catproc;

    var  taken : string[10];

    begin
      taken := 'Already Selected';
      repeat
        if count >= 6 then
        begin
          clear;
          lines(4);
          writeln('      No more categories available for special list.');
          writeln('              Please print target listing');
          lines(2);
          spacebar;
          exit(catproc);
        end;
        clear;
        case searcher of
          0 : ;
          1 : Tcheck := taken;
          2 : Ccheck := taken;
          3 : Scheck := taken;
          4 : Pcheck := taken;
```

```
      5 : Acheck := taken;
      6, 7 : statcheck := taken;
    end;
    catmenu;
    select;
    read(ch);
    if ch in charmenu then
    begin
      if eoln(input) then readln;
      case ch of
        '1' : begin
                if Icheck = taken then tellused := true
                else DBtype;
              end;
        '2' : begin
                if Scheck = taken then tellused := true
                else DBclass;
              end;
        '3' : begin
                if Scheck = taken then tellused := true
                else DBSnassen;
              end;
        '4' : begin
                if Pcheck = taken then tellused := true
                else DBpri;
             end;
        '5' : begin
                if Acheck = taken then tellused := true
                else DBacc;
              end;
        '6' : begin
                if statcheck = taken then tellused := true
                else DBstatus;
              end;
        'P','p','R','r','Q','q' : exit(catproc);
        '7','8','9' : menuerror;
        '?' : begin
                lines(1);
                writeln('      See prior menu for information');
                lines(1);
                spacebar;
               end;
      end
    end
    else menuerror;
  if tellused then
    begin
      clear;
      lines(5);
      writeln('      Category has already been selected. Please');
      writeln('      choose another category from the unused items');
      writeln('      on the menu listing. To start a new listing,');
      writeln('      choose option 7 to return to the main menu.');
      lines(3);
```

```
          searcher;
          telused := false;
        end;
      until tenloop = true;
    end;


procedure screenlist;

    var   ster : char;
          pribuff : string[2];
          sabuff : string[4];
          holdcar : char;
          pager : integer;


  procedure header;

        var  listing : string;

        begin
          listing := ' ';
          lines(1);
          writeln('                    SPECIAL TARGET LISTING');
          writeln('                    ----------------------');
          write('Categories:');
          if count > 8 then count := 8;
          for i := 1 to count do
            listing := concat(listing,' ',cat[i]);
          writeln(listing);
          lines(1);
          writeln('TGT NO   CL PRI  LOCATION   ALT  SAAFF  DESCRIPTION');
          writeln('------   -- ---  --------   ---- -----  -----------');
        end;


   begin{screenlist}
     clear;
     header;
     pribuff := '    ';
     sabuff := '    ';
     pager := 8;
     for i := 0 to reccount - 1 do
     begin
       ster := ' ';
       if database[i].flag = on then
       begin
         holdcar := database[i].pri;
         case holdcar of
           '1' : pribuff := 'I   ';
           '2' : pribuff := 'II  ';
           '3' : pribuff := 'III';
```

```
          '4' : pribuff := 'IN ';
        end;
        holdcar := database[i].sc;
        case holdcar of
          '1' : sabuff := 'ARTY';
          '2' : sabuff := 'NGF ';
          '3' : sabuff := 'AIR ';
          '4','5','6','7' : sabuff := 'COMB';
          '8' : sabuff := 'OTHR';
          '9' : sabuff := 'NONE';
        end;
if (database[i].stat = '1') or (database[i].stat = '2') then star := '*';
write(database[i].tnum,star,'  ',database[i].class,'  ',pribuff,'  ');
write(database[i].loc,'  ',database[i].alt:4,'  ',sabuff,'  ');
writeln(database[i].desc);
      pager := pager + 1;
      if pager = 20 then
      begin
        lines(1);
        spacebar;
        clear;
        lines(2);
        pager := 0;
      end;
    end;
  end;
  lines(1);
  writeln('     NOTE:  * indicates target list');
  lines(1);
  spacebar;
end;


      procedure reset;

      begin
        tellused := false;
        amount := 6;
        left := 8;
        count := 0;
        searcher := 0;
        first := true;
        Tcheck := emptystring;
        Ccheck := emptystring;
        Pcheck := emptystring;
        Acheck := emptystring;
        Scheck := emptystring;
        statcheck := emptystring;
        for i := 0 to reccount - 1  do
          database[i].flag := off;
        for i := 1 to 6 do
            cat[i] := emptystring;
      end;
```

```
procedure queryproc;

  begin
    count := 0;
    repeat
    clear;
    lines(2);
    writeln('              SPECIAL TARGET LISTING');
    writeln(dots);
    lines(2);
    writeln('       The options are:');
    lines(1);
    writeln('              1.  Form a new target listing');
    writeln('              2.  Continue to process');
    writeln('              3.  Write the special list to the screen');
    writeln('              4.  Information about this procedure');
    writeln('              5.',return);
    lines(1);
    select;
    read(ch);
    if ch in ['1','2','3','4','5','R','r','0'] then
    begin
      if eoln(input) then readln;
      case ch of
        '1' : begin
                reset;
                catproc;
              end;
        '2' : catproc;
        '3' : screenlist;
        '4','?' : begin
                    clear;
                    getfile('queryinfo.txt');
                    spacebar;
                  end;
        '5','R','r' : exit(query);
      end
    end
    else menuerror;
    until menulocp = true;
  end;



      begin {query}
        reccount := 0;
        emptystring :=  '                    ';
        searcher := 0;
charmenu := ['1','2','3','4','5','6','7','8','9','R','r','?','0','R','r',',','.'];
        getdatabase;
        reset;
        queryproc;
```

```
end;

(*.................END  CUBRY.......................*)
```

```
(*      BDA.TEXT     *)

     segment procedure targetmod;

     var
        stat1 : string[6];
        stat2 : string[3];
        ttypebuf : string[4];
        pribuf : string[6];
        sabuf : string[14];
        accbuf : string[9];
        duplicate, fetchback, outprocess, first : boolean;


     procedure fetchtgt(grid : integer); forward;
     procedure readin; forward;
     procedure checkDTG(var strng : string; var check:boolean); forward;
     procedure process; forward;
     procedure cutstring(strngsize : integer); forward;
     procedure putinfile; forward;


segment procedure newBDA;


procedure DTGofBDA;


begin
  currentgt.BDA[BDAcounter].DTGsurv := '        ';
  while not finished do
    begin
      clear;
      lines(6);
      writeln('     ENTER DTG TARGET WAS ATTACKED...6 digits and 1 letter.');
      prompt;
      readin;
      checkDTG(str,ok);
      if quit then exit(DTGofBDA);
      if helpme or not ok then
        begin
          finished := false;
          lines(1);
          getfile('dtgofbda.text');
          lines(1);
          returnbar;
        end;
        if  (flg = 2) and finished  then
        currentgt.BDA[BDAcounter].DTGsurv := str;
      end;
    end;
```

```
     procedure firingunit;

begin
  currentet.BDA[BDAcounter].fireunit := '        ';
  while not finished do
   begin
    lines(2);
    writeln('      ENTER FIRING UNIT....do not exceed 8 characters');
    prompt;
    readln;
    if helpme then
    begin
      finished := false;
      lines(1);
      getfile('funit1.text');
      lines(1);
      returnbar;
    end
    else if length(str) > 8 then
      begin
        lines(1);
        getfile('funit2.text');
        lines(1);
        finished := false;
        currentet.flag[n] := on;
        returnbar;
      end
    else if quit then exit(firingunit)
    else if (flg = 2) and finished  then
    currentet.BDA[BDAcounter].fireunit := str;
  end;
end;


     procedure rounds;

begin
  currentet.BDA[BDAcounter].ntrnds := '            ';
  while not finished do
    begin
    lines(2);
    writeln('      ENTER NUMBER AND TYPE OF ROUNDS FIRED');
    prompt;
    readln;
    if helpme then
    begin
      finished := false;
      lines(1);
      getfile('rounds.text');
      lines(1);
      returnbar;
    end
    else if length(str) > 10 then
      begin
```

```
        lines(1);
        getfile('rounds1.text');
        lines(1);
        finished := false;
        currentgt.flag[n] := on;
        returntar;
      end
      else if quit then exit(rounds)
      else if (flg = 2) and finished then
      currentgt.BDA[BDAcounter].ntrnds := str;
    end;
end;


    procedure damagemenu( param : integer);

    var  kind : string[8];

    begin
      if param = 1 then kind := 'REPORTED'
      else kind := 'ASSESSED';
      clear;
      lines(5);
      writeln('      ENTER DAMAGE ',kind);
      lines(1);
      getfile('damagemen.text');
      lines(2);
    end;



    procedure damagrept;

  begin
      currentgt.BDA[BDAcounter].damrep := '9';
      repeat
        damagemenu(1);
        select;
        read(cn);
        if cn in ['1'..'8'] then
        begin
          if eoln(input) then readln;
          finished := true;
          currentgt.BDA[BDAcount].damrep := cn;
        end
        else if eoln(input) then exit(damagrept)
        else if cn in ['C','q'] then
        begin
          quit := true;
          exit(damagrept);
        end
        else if cn = '?' then
        begin
          lines(1);
```

```
          getfile('damrep.text');
          lines (1);
          returnbar;
        end
      else menuerror;
    until finished = true;
  end;


   procedure damagassd;

   begin
     currentgt.BDA[BDAcounter].damass := '9';
     repeat
       damagemenu(2);
       select;
       read(ch);
       if ch in ['1'..'8'] then
       begin
         if eoln(input) then readln;
         finished := true;
         currentgt.BDA[BDAcount].damass := ch;
       end
       else if eoln(input) then exit(damagassd)
       else if ch in ['Q','q'] then
       begin
         quit := true;
         exit(damagassd);
       end
       else if ch = '?' then
       begin
         lines(1);
         getfile('damass.text');
         lines(1);
         returnbar;
     end
   else menuerror;
  until finished = true;
  end;


procedure BDAremarks;
var  x : integer;

begin
  currentgt.BDA[BDAcounter].BDAtext := nostring;
  while not finished do
  begin
    clear;
    lines(6);
    writeln('      ENTER BDA....do not exceed one line');
    prompt;
    readln;
```

```
    if helpme then
    begin
      finished := false;
      lines(1);
      getfile('bdarem.text');
      lines(1);
      returnbar;
    end;
    if quit then exit(BDAremarks);
    if  (flg = 2) and finished then
     begin
       buffer := '                                        ';
       outstring(40);
       currentgt.BDA[BDAcount].BDAtext := str;
     end;
  end;
end;



     procedure BDAinfo;

     begin
       clear;
       lines(2);
       writeln('                    BATTLE DAMAGE ASSESSMENT');
       lines(2);
       writeln('      For information on adding a target surveillance');
       writeln('                to the target file, type a ?.');
       lines(3);
       writeln('         **   To continue, press the RETURN key.   **');
       prompt;
       read(cn);
       if cn = '?' then
         begin
           clear;
           lines(2);
           getfile('bdainfo.text');
           lines(3);
           spacebar;
           clear;
         end;
      end;



     begin {newBDA}
       if not current then
       begin
         BDAinfo;
         retcntgt(1);
```

```
    if quit then exit(newBDA);
    i := 1;
    while (currentgt.flag[16 + i] = off) do
      begin
        i := i + 1;
        if i = 4 then
        begin
          i := 1;
          currentgt.flag[16 + i] := on;
        end;
      end;
    BDAcounter := i;
    n := 16 + i;
  end;
  endBDA := false;
  while not endBDA do
  begin
    for ii := 1 to 6 do
    begin
        finished := false;
        case ii of
          1 : BTGorBDA;
          2 : firingunit;
          3 : rounds;
          4 : damagrept;
          5 : damagassd;
          6 : begin
                BDAremarks;
                endBDA := true;
              end;
        end;
    end;
    if current then exit(newBDA)
    else begin
        with currentgt do
          begin
            if stat = '2' then stat := '1'
            else if stat = '4' then stat := '3'
            else if stat = '6' then stat := '5'
          end;
        with currentBT do
          begin
            if stat = '2' then stat := '1'
            else if stat = '4' then stat := '3'
            else if stat = '6' then stat := '5'
          end;
        putinfile;
    end;
end;
```

```
(*        TGTPROCS.TEXT     *)

(*.............................................................*)


   procedure mandmsg;

   begin
     lines(1);
     getfile('mandmsg.text');
     lines(1);
     spacebar;
   end;


   procedure readin;

   var  len : integer;

   begin
   helpme := false;
   readln(str);
   len := length(str);
   if len = 0 then flg := 0
   else if str[1] = '?' then flg := 1                          {help}
   else if (len = 1) and (str[1] in {'Q','q'}) then flg := 3 {quit}
   else flg := 2;                                              {continue}
   case flg of
     0 : begin
         if mandatoryitem then mandmsg
         else finished := true;
         end;
     1 : helpme := true;
     2 : begin
           currentat.flag[n] := off;
           finished := true;
         end;
     3 : quit := true;
       end
     end;



     procedure checknum(var string : string ; var check : boolean);

     var x,i : integer;

     begin
       check := true;
```

```
        x := length(string);
        if x = 0 then
          begin
            fetchback := true;
            exit(checknum);
          end;
        if x <> 6 then
        begin
          check := false;
          exit(checknum);
        end;
        for i := 1 to 2 do
        if not (string[i] in ['A'..'Z']) then
          begin
            check := false;
            writeln('    Use upper case letters for target designator');
            exit(checknum);
          end;
        for i := 3 to 6 do
          if not (string[i] in ['0'..'9']) then check := false;
        end;



procedure checkdigit (var string:string; var check : boolean; rng : integer );

var  i, x : integer;

begin
  check := true;
  x := length(string);
  if x = 0 then
          begin
            fetchback := true;
            exit(checkdigit);
          end;
  if x <> rng then
  begin
    check := false;
    exit(checkdigit);
  end;
  for i := 1 to rng do
    if not (string[i] in ['0'..'9']) then check := false;
end;


procedure outstring ;

(*   (stringsize : integer) removed for fwd dec    *)
```

```
var    cutter : integer;

begin
  if length(str) > strngsize then
  begin
    for cutter := 1 to strngsize do
      buffer[cutter] := str[cutter];
    str := buffer;
  end;
end;




   procedure checkDTG ;


   (*     (var string : string; var check : boolean) removed for two lec *

   var   i : integer;

      begin

     check := true;
     if length(str) - 7 then exit(checkDTG);
     if length(string) <> 7 then
     begin
       check := false;
       exit(checkDTG);
     end;
     for i := 1 to 5 do
       if not(string[i] in ['0'..'9']) then
       begin
         check := false;
         exit(checkDTG);
       end;
     if not (string[7] in ['A'..'Z']) and not (string[7] in ['a'..'z']) then
       check := false;
   end;

  (*.............................................................*)
```

```
{*   [illegible]   *}

procedure verifyend;

begin
  for i := [illegible] do
    if [illegible] = '[illegible]' then
    begin
      [illegible];
      writeln('[illegible]');
      [illegible];
      writeln('[illegible]');
      writeln('[illegible]');
      writeln('[illegible]');
      [illegible];
      [illegible];
      [illegible] := true;
      [illegible] := true;
    end;
end;


[illegible] [illegible];

begin
  while not [illegible] do
  begin
    [illegible];
    [illegible];
    writeln('[illegible]');
    [illegible];
    [illegible];
    if [illegible] then [illegible];
    [illegible];
    if keyp[illegible] or [illegible] then
    begin
      finished := false;
      lines(1);
      writeln('[illegible]');
      lines(1);
      return(2);
    end;
    if (ch = [illegible]) and finished then
    begin
      [illegible] := true;
      verifyend;
      if [illegible] then
        begin
          [illegible] := true;
          [illegible];
        end;
```

```
procedure table;

begin
  [illegible];
  while not finish do
    begin
      clrscr;
      lines(0);
      writeln('      [illegible]      lines');
      prompt;
      readln;
      checkquit(ch,quit);
      if quit then exit(table);
      if length(ch) <> 0 then
        begin
          finished := false;
          lines(1);
          rewrite('table.text');
          lines(1);
          [illegible];
        end;
        if (ch <> '') and finished then
          begin
            [illegible] := str;
            [illegible] := str;
          end;
      end;
    end;


procedure tabsec;

begin
  while not finish do
  begin
    clrscr;
    lines(0);
    writeln('      [illegible]');
    prompt;
    readln;
    if quit then exit(tabsec);
    if [illegible] then
```

```
    begin
      finished := false;
      rlines(1);
      getline('testsc.text');
      rlines(1);
      returnlar;
    end;
    if (ch = 'z') and finished then
    begin
      buffer := '                                                    ';
      substring(n);
      currentst.d[n] := st;
      buffer := '                                    ';
      substring(n);
      currentl.dsc := st;
    end;
  end;
end;


procedure tstclass;

begin
  repeat
  clear;
  rlines(6);
  getline('classtemp.text');
  rlines(1);
  select;
  read(ch);
  if ch in ['a'..'c'] then
  begin
    if eoln(input) then readln;
      currentst.tag[1] := ch;
      finished := true;
    case ch of
      'a' : begin
              currentst.class := 'a';
              currentwl.class := 'a';
            end;
      'b' : begin
              currentst.class := 'b';
              currentwl.class := 'b';
            end;
      'c' : begin
              currentst.class := 'c';
              currentwl.class := 'c';
```

```
        end;
    'a' : begin
            currentmt.class := 'a';
            currentmt.class := 'a';
        end;
    'b' : begin
            currentmt.class := 'b';
            currentmt.class := 'b';
        end;
    'c','f','i' : begin
                    currentmt.line[n] := [illegible];
                    finished := true;
                    [illegible];
                  end;
    'x' : begin
            clear;
            lines(1);
            readfile('lines.text');
            lines(1);
            [illegible];
            currentmt.line[n] := [illegible];
            finished := false;
        end
    end
  end
  else if [illegible] in ['+','-'] then
        begin
          quit := true;
          exit(tgtclass);
        end
  else if (eoln(input)) then [illegible]
  else menuerror;
  until finished = true;
end;



procedure tmpfl;

begin
  repeat
  clear;
  lines(4);
  readfile('tgtprimenu.text');
  lines(1);
  select;
  read(ch);
  if ch in tenucar then
  begin
    if eoln (input) then readln;
    if ch in ['1'..'4'] then
    begin
      currentmt.line[n] := off;
```

```
      finished := true;
      currentgt.pri := ch;
      currentgT.pri := ch;
    end;
  if ch = 'Y' then
     begin
       clear;
       lines(1);
       getfile('priority.text');
       lines(3);
       spacebar;
     end;
  if ch in ['b','B','R','r'] then menuerror;
  end
  else if ch in ['Q','q'] then
         begin
           quit := true;
           exit(tatpri);
         end
  else if (eoln(input)) then menuerr
  else menuerror;
  until finished = true;
end;


  procedure tetstatus;

  var loop : boolean;
  code : integer;


  procedure active;

  begin
    loop := false;
    repeat
      clear;
      lines(5);
      writeln('      ENTER TARGET STATUS--ACTIVITY');
      lines(1);
      writeln('       The options are:');
      lines(1);
      writeln('           1.  Active');
      writeln('           2.  Inactive');
      lines(1);
      writeln('      PLEASE ENTER OPTION NUMBER AND PRESS RETURN');
      prompt;
      read(ch);
      if ch in menuset then
      begin
        if not eoln(input) then readln;
        if ch = '1' then
        begin
          code := 0;
```

```
            loop := true;
          end;
          if ch = '2' then
          begin
            code := 12;
            loop := true;
          end;
          if ch = '3' then
          begin
            lines(1);
writeln('    FR active target is the which is first in the input list of ');
writeln('the list of targets. An inactive target is in the  editor.');
            lines(1);
            spacetab;
          end;
          if ch in ['3','4','5','6','7','8'] then begin
          end
          else if ch in ['e','q'] then
          begin
            quit := true;
            exit(s list);
          end
          else if (ch=[r,p]) then begin
          else begin end;
        until loop = true;
    end;

    procedure listel;

    begin
      loop := false;
      repeat
        clrs;
        lines(1);
        writln('        Field Artillery Simulation Editor');
        lines(1);
        writln('        The options are:');
        lines(1);
        writln('          1.  Input list');
        writln('          2.  List of targets');
        lines(1);
        select;
        read(ch);
        if ch in [a,m,d,t] then
        begin
          if ch=[r,p] then read(ch);
          if ch = '1' then
          begin
            code := code + 1;
            loop := true;
          end;
          if ch = '2' then
          begin
            code := code + 2;
```

```
            loop := true;
          end;
          if ch = 'X' then
          begin
            lines(2);
writeln('The target list refers to the current list of the threat');
writeln('subsystems. If a target is not on the current list then it');
writeln('                is on the list of threats.');
            lines(1);
            spinset;
          end;
          if ch in [' ','-','<','>',' ','/'] then [illegible] ,
          [illegible]
          else if ch in ['e','d'] then
               begin
                 quit := true;
                 exit(list-[illegible]);
               end
      else if (eoln(input)) then [illegible]
      else [illegible];
    until loop = true;
    end;

    procedure attacker;

    begin
    loop := false;
    repeat
        clear;
        lines(2);
        writeln('      ENTER TARGET STATUS-SUBSYSTEM AFTER ATTACK ');
        lines(1);
        writeln('       The options are:');
        lines(1);
        writeln('           1. Attacked');
        writeln('           2. Not attacked');
        lines(1);
        select;
        read(ch);
        if ch in [enter] then
        begin
          if eoln(input) then [illegible];
          if ch = '1' then
          begin
            [illegible] := [illegible] + 7;
            loop := true;
          end;
          if ch = '2' then
          begin
            [illegible] := [illegible] + 8;
            loop := true;
          end;
          if ch = '.' then
          begin
```

```
            flush(output);
writeln('Attached targets are those attached to operations area');
writeln('for which there is a qualified or trained operator.');
            resetin;
            spaceout;
          end;
          if ch in ['?','4','5','1','2','3'] then fnd:=false;
          end
          else if ch in ['q','Q'] then
              begin
                quit := true;
                exit(attachment);
              end
          else if (eoln(input)) then readln
          else writeln(error);
      until loop = true;
    end;


    begin        {tstatus}
      repeat
      flush(output);
      active;
      if quit then exit(tstatus);
      listen;
      if quit then exit(tstatus);
      attached;
      if quit then exit(tstatus);
      finished := false;
      case code of
       17 : ch := '1';
       18 : ch := '2';
       19 : ch := '3';
       20 : ch := '4';
    21, 24 : begin
              flush(output);
              writeln('     Coordination not possible....please reenter status');
              spaceout;
              finished := false;
            end;
       26 : ch := '5';
       27 : ch := '6';
       end;
       until finished;
       if ch in ['1','5','6'] then nbtlinks := nbtlinks - 1;
       currentgt.flag[i] := ch;
       footmate.stat := ch;
       currentgt.stat := ch;
      end;
```

```
procedure tttype;
var   [illegible] : [illegible];

begin
  [illegible] := ['1','2','3','4','5','6','7','8','9'];
  repeat
    clear;
    lines(2);
    settitle('tttypetest.text');
    lines(1);
    select;
    read(c);
    if ch in [illegible] then
    begin
      if eln(input) then readln;
      finished := true;
    end
    else if c in ['e','q'] then
    begin
      quit := true;
      exit(tttype);
    end
    else if (eln(input)) then
    begin
      clear;
      lines(2);
      [illegible];
    end
    else if c = '?' then
    begin
      lines(1);
      settitle('tttype.text');
      lines(1);
      returnter;
    end
    else returnter;
  until finished;
  currentgt.ttype := ch;
  currentgt.line[k] := ch;
  currentGT.ttype := ch;
end;


procedure tetrit;

    var x, y : integer;


begin
  currentgt.alt := nostring;
  currentGT.alt := nostring;
  while not finished do
  begin
```

```
    clear;
    lines(1);
    writeln('      [illegible]');
    rewrite;
    readln;
    if quit then exit(bigbrk);
    s := length(str);
    for i := 1 to s do
      if not (str[i] in ['0'..'9']) then [illegible]
    if s > 4 then [illegible] else [illegible];
    if neight then
    begin
      finished := false;
      lines(1);
      getfile('textit.text');
      lines(1);
      return[illegible];
    end;
    if (lip = 2) and finished then
    begin
      [illegible].exit := str;
      [illegible].exit := str;
    end;
  end;
end;


procedure checkset;

var   [illegible] : [illegible];

begin
  [illegible] := ['1','2','3','4','5','6','7','8','9'];
    repeat
      clear;
      lines(2);
      getfile('menu.text');
      lines(1);
      select;
      read(ch);
      if ch in [illegible] then
      begin
        if odd([illegible]) then [illegible];
        finished := true;
      end
      else if ch in ['0','q'] then
      begin
        quit := true;
        exit(checkset);
      end
```

```
      else if ch = '?' then
        begin
          clear;
          lines(5);
          getfile('sa.text');
          lines(1);
          returnter;
        end
        else if eoln(input) then
        begin
          currentgt.sa := 'y';
          currentGT.sa := 'y';
          exit(SAassem);
        end
        else menuerror;
      until finished;
      currentgt.sa := ch;
      currentgt.flag[n] := off;
      currentGT.sa := ch;
  end;


procedure remarks;

begin
  currentgt.rem := postrim;
  while not finished do
  begin
    clear;
    lines(5);
    writeln('      ENTER REMARKS CONCERNING TARGET....DO NOT EXCEED ONE LINE');
    prompt;
    readln;
    if helpme then
    begin
      finished := false;
      lines(1);
      getfile('remarks.text');
      lines(1);
      returnter;
    end;
    if quit then exit(remarks);
    if (flag = 2) and finished then
    begin
      buffer := '                                        ';
      cutstring(40);
      currentgt.rem := str;
    end;
  end;
end;
```

```
procedure tapref;

begin
  currentgt.mapprefer := nostring;
  while not finished do
  begin
    clear;
    lines(6);
    writeln('      ENTER TARGET MAP REFERENCE....do not exceed 20 characters.'
    prompt;
    readln;
    if helpme then
    begin
      finished := false;
      lines(1);
      getfile('mapref.text');
      lines(1);
      returnter;
    end;
    if quit then exit(mapref);
    if (fle = 2) and finished then
    begin
      buffer := '                    ';
      outstring(20);
      currentgt.mapprefer := str;
    end;
  end;
end;


procedure source;

begin
  currentgt.sour := nostring;
  while not finished do
  begin
    clear;
    lines(6);
    writeln('      ENTER SOURCE OF TARGET....do not exceed 20 characters.' ;
    prompt;
    readln;
    if helpme then
    begin
      finished := false;
      lines(1);
      getfile('sour.text');
      lines(1);
      returnter;
    end;
    if quit then exit(source);
    if (fle = 2) and finished then
    begin
      buffer := '                    ';
```

```
      cutstring(2);
      currentgt.sour := str;
    end;
  end;
end;


procedure afotonum;

begin
  currentgt.photonum := nostring;
  while not finished do
  begin
    clear;
    lines(6);
    writeln('      ENTER AERIAL PHOTO NUMBER');
    prompt;
    readln;
    if helpme then
    begin
      finished := false;
      lines(1);
      getfile('afotonum.text');
      lines(1);
      returnter;
    end;
    if quit then exit(afotonum);
    if (flg = 2) and finished then
    begin
      buffer := '                    ';
      cutstring(15);
      currentgt.photonum := str;
    end;
  end;
end;

        procedure photogrid;

        begin
 range := 5;
 while not finished do
 begin
   clear;
   lines(6);
   getfile('photogrid1.text');
   prompt;
   readln;
   if flg = 0 then
   begin
     currentgt.photocord := nostring;
     exit(photogrid);
   end;
   if (flg = 2) and (length(str) = 1) and (str[1] in ['S','s']) then
```

```
      begin
        currentet.photocord := currentet.loc;
        exit(photogrid);
      end;
    checkulait (str,ok,raise);
    if quit then exit (photogrid);
    if helpme or not ok then
    begin
      finished := false;
      lines(1);
      getfile('photogrid2.text');
      lines(1);
      returnbar;
    end;
    if (fig = 2) and finished then currentet.photocord := str;
  end;
end;


        procedure DTGactive;

begin
  currentet.DTGact := nostring;
  while not finished do
    begin
      clear;
      lines(6);
      writeln('      ENTER TTG TARGET WAS ACTIVATED...6 digits and 1 letter.');
      prompt;
      readln;
      check TG(str,ok);
      if quit then exit(DTGactive);
      if helpme or not ok then
        begin
          finished := false;
          lines(1);
          getfile('itgact.text');
          lines(1);
          returnbar;
        end;
        if (fig = 2) and finished then currentet.DTGact := str;
      end;
    end;



  procedure tgtaccuracy;

  begin
    currentet.acc := '4';
    currentCT.acc := '4';
    repeat
    clear;
```

```
  lines(4);
  getfile('tetaccmenu.text');
  lines(1);
  select;
  read(ch);
  if ch in menuchar then
  begin
    if eoln (input) then readln;
    if ch in ['1'..'4'] then finished := true;
    if ch in ['b','B','R','r'] then menuerror;
    if ch = '?' then
    begin
      lines(1);
      getfile('tetacc.text');
      lines(1);
      returnmar;
    end;
   end
   else if ch in ['Q','q'] then
        begin
          quit := true;
          exit(tetaccuracy);
        end
  else if eoln(input) then exit(tetaccuracy)
  else menuerror;
  until finished = true;
      currentet.flag[n] := off;
      currentet.acc := ch;
      currentCT.acc := ch;
end;
```

```
            (*      CHANGE PROCEDURE      *)

   procedure caseproc; forward;
   procedure displaytgt; forward;


procedure change;

var  punchout, yes : boolean;


procedure changeinfo;

begin
  clear;
  lines(2);
  getfile('changeinfo.text');
  lines(2);
  spacebar;
  clear;
end;


             procedure yescheck;

   begin
     finished := false;
     yes := false;
     lines(2);
     writeln('      Change Y  Y(es)  N(o)');
     prompt;
     read(ch);
     if (ch = 'Y') or (ch = 'y') then yes := true
     else if ( ch = 'o') or ( ch = 'q') then punchout := true;
   end;


   procedure chgproc2;

begin
  with currentgt do
    begin
      clear;
      lines(8);
      writeln('  Current air photo grid is...',photogrid);
      yescheck;
      if yes then photogrid;
      if punchout then exit(chgproc2);
      clear;
      lines(8);
```

```
      writeln('  Current SIG target activated is...',sigact);
      yescheck;
      if yes then ITGactive;
      if punchout then exit(changeproc2);
      clear;
      lines(5);
      writeln('  Current accuracy is...',accuracy);
      yescheck;
      if yes then tgtaccuracy;
      if punchout then exit(changeproc2);
    end;
end;


procedure changeproc;

begin
  quit := false;
  caseproc;
    with currentgt do
      begin
        clear;
        lines(2);
        writeln('        Target  number  ',tnum);
        lines(5);
        writeln('  Current location is...',loc);
        yescheck;
        if yes then tgtloc;
        if punchout then exit(changeproc);
        clear;
        lines(4);
        writeln('  Current description is...',desc);
        yescheck;
        if yes then tgtdesc;
        if punchout then exit(changeproc);
        clear;
        lines(4);
        writeln('  Current class is...',class );
        yescheck;
        if yes then tgtclass;
        if punchout then exit(changeproc);
        clear;
        lines(4);
        writeln('  Current priority is...',priority);
        yescheck;
        if yes then tgtpri;
        if punchout then exit(changeproc);
        clear;
        lines(4);
        writeln('  Current status is...',stat1);
        writeln('      On target list?..',stat2);
        yescheck;
        if yes then tgtstatus;
        if punchout then exit(changeproc);
```

```
        clear;
        lines(5);
        writeln('  Current type is...',ttypeout);
        yescheck;
        if yes then tettype;
        if punchout then exit(changeproc);
        clear;
        lines(5);
        writeln('  Current altitude is...',alt);
        yescheck;
        if yes then tgtalt;
        if punchout then exit(changeproc);
        clear;
        lines(5);
        writeln('  Current supporting arr assigned is...',satot);
        yescheck;
        if yes then SAassign;
        if punchout then exit(changeproc);
        clear;
        lines(5);
        writeln('  Current remarks are...',rem);
        yescheck;
        if yes then remarks;
        if punchout then exit(changeproc);
        clear;
        lines(5);
        writeln('  Current map reference is...',mapreter);
        yescheck;
        if yes then mapref;
        if punchout then exit(changeproc);
        clear;
        lines(5);
        writeln('  Current source is...',sour);
        yescheck;
        if yes then source;
        if punchout then exit(changeproc);
        clear;
        lines(5);
        writeln('  Current air photo number is...',photonum);
        yescheck;
        if yes then airotonum;
        if punchout then exit(changeproc);
        chgproc2;
    end;
  end;



begin{change}
  mandatoryitem := false;
  changeinfo;
  if not current then retcntgt(1);
```

```
if quit then exit(change);
repeat
  punchout := false;
  clear;
  lines(2);
  writeln('      Target  ',currentgt.thum,' is loaded into memory');
  lines(2);
  getfile('changemenu.text');
  lines(2);
  select;
  read(ch);
  if ch in ['1','2','3','4','R','r','Y'] then
  begin
    if eoln(input) then readln;
    case ch of
      '1' : displaytgt;
      '2' : changeproc;
      '3' : begin
              putinfile;
              outprocess := true;
              exit(change);
            end;
      '4','R','r' : exit(change);
      'Y' : begin
              lines(2);
              getfile('changex.text');
              lines(2);
              spacebar;
            end;
    end;
  end;
  until menuloop = true;
end;
```

```
{*****************************************************************}

                    {*    TARGET.TEXT    *}

  procedure bulletin;

  var  j : integer;

  begin
    j := 0;
    i := 0;
    while not eof(target) do
    begin
        seek(target,i);
        get(target);
        bulletin[i] := target^.text .lin;
        i := i + 1;
        j := j + 1;
        if j = 10 then
          begin
            write(dot);
            j := 0;
          end;
      end;
  end;


  procedure initialize;

  var  noiser, n : integer;
       first : boolean;

begin
  first := true;
  i := 0;
  range := 0;
  fetchback := false;
  finished := false;
  quit := false;
  mandatoryitem := false;
  n := 1;
  while not finished do
  begin
    lines(0);
    if grid = 2 then writeln('      ENTER GRID LOCATION')
    else writeln('      ENTER TARGET NUMBER');
    prompt;
    readln;
    if grid = 2 then checkgrid (str,ok,range)
    else checknum(str,ok);
    if quit then exit(initialize);
    if fetchback then goto := true;
    if helpme or not ok then
```

```
begin
  finished := false;
  lines(2);
  if grid = 2 then
  begin
  getfile('latlon.text');
  end
  else
  begin
  getfile('tmp.text');
  end;
  lines(2);
  returnbar;
end;
if (rid = 2) and finished then
begin
  clear;
  lines(8);
  if grid = 2 then
  begin
    write('   Searching for grid coordinates   ',str);
    buildarrmap;
  end
  else write('  Searching for target   ',str);
  recnum := 1;
  write(dot);
  if grid = 2 then
    begin
      for recnum := 2 to numbertot - 1 do
      begin
        if arrmap[recnum] = str then
        begin
          lines(1);
write(lst(                  target no. ',trtmp[recnum],' has coordinates ',str);
          if first then gotoxy(1,wrk);
          first := false;
          r := r + 1;
        end;
      end;
      if r = 1 then recnum := normr;
      if r > 1 then
      begin
        finished := false;
        lines(1);
      writeln('     select the desired target number from the above list ');
        lines(1);
        spacebar;
        exit(r-trunc);
      end
  end
  else while (trtmp[recnum] <> str) and (r = 1) and (recnum < numbertot - 1) do
      recnum := recnum + 1;
    write(dot);
    if recnum = numbertot - 1 then
```

```
      begin
        finished := false;
        error;
      end
      else
      begin
        write(dot);
        seek(target,recnum);
        get(target);
        write(dot);
        currentpt := target^.textptr;
        seek(cf,recnum);
        get(cf^);
        current.l := cf^.currentl;
        write(dot);
      end;
    end;
  end;
end;


    procedure del;

    begin
      cf^.overflow := emptyvbuf;
      seek(cf,recnum);
      put(cf^);
      target^.textptr := emptytext;
      seek(target,recnum);
      put(target);
      textsb[recnum] := 'xxxxxx';
    end;


    procedure deletetext;

    begin
      clear;
      lines(2);
      writeln('                    delete text');
      lines(2);
      getfile('deltxt.text');
      retcontext(1);
      if not quit then del;
    end;




    procedure cas-proc;

    var  noinsert : char;
```

```
begin
  with [illegible] do
  begin
    [illegible] := [illegible];
    case [illegible] of
      '1' : [illegible] := 'AFI                    ';
      '2' : [illegible] := 'VOF                    ';
      '3' : [illegible] := 'AIA                    ';
      '4' : [illegible] := 'AIA, AFI               ';
      '5' : [illegible] := 'AIA, VOF               ';
      '6' : [illegible] := 'AFI, VOF               ';
      '7' : [illegible] := 'AIA, AFI, VOF          ';
      '8' : [illegible] := 'OTHER                  ';
      '9' : [illegible] := 'NOA                    ';
    end;
    [illegible] := [illegible];
    case [illegible] of
      '1' : [illegible] := 'CONFIRMED  ';
      '2' : [illegible] := 'PROBABLE   ';
      '3' : [illegible] := 'POSSIBLE   ';
      '4' : [illegible] := 'UNKNOWN    ';
    end;
    [illegible] := [illegible];
    case [illegible] of
      '1' : [illegible] := '1   ';
      '2' : [illegible] := '1A  ';
      '3' : [illegible] := '1AA ';
      '4' : [illegible] := '1B  ';
    end;
    [illegible] := [illegible];
    case [illegible] of
      '1' : [illegible] := '[illegible]';
      '2' : [illegible] := '[illegible]';
      '3' : [illegible] := '[illegible]';
      '4' : [illegible] := '[illegible]';
      '5' : [illegible] := '[illegible]';
      '6' : [illegible] := '[illegible]';
      '7' : [illegible] := '[illegible]';
      '8' : [illegible] := '[illegible]';
      '9' : [illegible] := '[illegible]';
    end;
    [illegible] := [illegible];
    case [illegible] of
      '1','2' : begin
                  [illegible] := '[illegible]';
                  [illegible] := '[illegible]';
                end;
      '3','4' : begin
                  [illegible] := '[illegible]';
                  [illegible] := '[illegible]';
                end;
      '5','6' : begin
                  [illegible] := '[illegible]';
```

```
                    stat? : ')) ';
                  end;
            end;
          end;
        end;


      procedure telnoard;


      begin
        clear;
        with currentat do
          begin
writeln(dots);
writeln('                    : [head: ],[trn,]');
writeln('                    ----------');
writeln('Location: ',loc,'      alt: ',alt,'          type: ',tltcatt );
writeln('Description: ',desc);
writeln('Class    : ',class,'                    status: ',status);
writeln('Priority : ',priorit,'                Ini Instr: ',instr,);
writeln('SA assng : ',satur);
writeln('Source of Info: ',source);
writeln('LTG activated: ',ltgact),
writeln('Photo No: ',photono,'          Tape Ref: ',taperef);
writeln('Photo Coord: ',photocoord,'    Assigned: ',assgnd),
writeln('Remarks: ',rmk),
          end;
      end;



procedure zlAgain;

var holder : char;
    tempbuf : string[11];
    tmassbuf : string[11];

      begin
        tmassbuf := '           ';
        tempbuf := '           ';
writeln('--------------------SUMMARY------------------------------------');
writeln('              Firing      Vehicle          Number       Target');
writeln('     TIS      Unit        Points           Expended     Assigned');
writeln('     ---      ------      -------          --------     --------');
        for i := 1 to 6 do
          if (currentat.fire[i + 1] <> ' ') and (currentat.ZF[i]
            begin
              holder := tempbuf;
              case holder of
              '1' : tempbuf := 'DAVAGE    ',
              '2' : tempbuf := 'DESTROYED ';
```

```
        '6' : damreptut := 'INSTRUCTIONS';
        '4' : damreptut := 'PARAMETER';
        '5' : damreptut := 'DESTABILIZER';
        '0' : damreptut := 'INSTRUCTIONS';
        '7' : damreptut := 'NOISEMAKER';
        'a' : damreptut := 'UNABLE';
        'e' : damreptut :=
        end;
        dcicdar := damass;
        case dcicdar of
        '1' : damasstut := 'PARAMETER';
        '2' : damasstut := 'DESTABILIZER';
        '3' : damasstut := 'INSTRUCTIONS';
        '4' : damasstut := 'PARAMETER';
        '5' : damasstut := 'DESTABILIZER';
        '6' : damasstut := 'INSTRUCTIONS';
        '7' : damasstut := 'NOISEMAKER';
        'a' : damasstut := 'UNABLE';
        'e' : damasstut :=
        end;
writeln(' ',Dsdpvir,'    ',[illegible],'    ',[illegible]);
writeln('        ',[illegible],'    ',damasstut);
writeln('sla: ',zbatent);
    end;
  end;


  procedure [illegible];

    begin
      with [illegible] do
        begin
          [illegible];
          [illegible];
          [illegible] ('1','2','3') [illegible];
        end;
      writeln('.........',[illegible],'    .........',
      repeat
        [illegible];
      until eoln(input);
    end;


  procedure [illegible];

  begin
    current := [illegible];
    finished := false;
    dmit := first;
    repeat;
      [illegible];
      lines(2);
      writeln(                    [illegible]
```

```
    [illegible]
    [illegible]
    [illegible]
```

```
procedure Access;

begin
  if [illegible] then writeln(pt[illegible]);
  repeat
    clrscr;
    writeln(title);
    lines(3);
    writeln('      Input [illegible] [illegible]');
    lines(2);
    writeln('      Edit Option [illegible]');
    lines(1);
    writeln('            1. [illegible] [illegible]');
    writeln('            2. [illegible] [illegible]');
    writeln('            3. [illegible] [illegible]');
    writeln('            4. [illegible] [illegible]');
    if [illegible] '1'..'[illegible]' [illegible]
    writeln('            5. [illegible] [illegible]');
    lines(1);
    [illegible];
    [illegible];
    if [illegible] in ['1'..'5'] then
    begin
      if [illegible] then [illegible];
      finished := true;
      case [illegible] of
        '1' : [illegible];
        '2' : begin
                finished := false;
                display;
              end;
        '3' : begin
                current := true;
                change;
                if [illegible] then [illegible];
              end;
        '4' : [illegible];
        '5' : if [illegible] = '1' or [illegible] then
                begin
                  [illegible] := 1;
                  current := true;
                  [illegible];
                  finished := false;
                end
              else [illegible];
      end;
    end
    else if [illegible] = '9' then
        begin
          finished := false;
```

```
          clear;
          lines(2);
          [illegible]('[illegible].text');
          lines(1);
          space[illegible];
        end
    else [illegible];
  until finished = true;
end;


    procedure [illegible];

    var count : integer;

    begin
      count := 0;
      for [illegible] := 1 to [illegible] do
        if [illegible] then count := count + 1;
      if count = 0 then
      begin
        clear;
        lines(2);
        writeln('   [illegible]', [illegible], ' [illegible]');
        lines(2);
        space[illegible];
        done := true;
      end
      else begin
              [illegible];
              lines([illegible]);
writeln('[illegible] have not been [illegible] target [illegible]');
              [illegible];
              lines(2);
              writeln('       The options are:');
              lines(1);
              writeln('         1.  Continue working on [illegible]');
              writeln('         2.  Stop working on target [illegible]');
              writeln('         3.  [illegible] changes to the target [illegible]');
              lines(1);
              space;
              read(ch);
              if ch in ['1','2','3','4'] then
                begin
                  if eoln(input) then readln;
                  case ch of
                    '1' : exit([illegible]);
                    '2' : begin
                            done := true;
                            exit([illegible]);
                          end;
                    '3' : begin
                            done := true;
```

```
                        current := true;
                        [illegible];
                        [illegible];
                        end;
                  '?' : begin
                          clear;
                          lines(5);
                          [illegible]('[illegible].text');
                          lines(2);
                          [illegible];
                        end
                  end
              end
              else [illegible];
              clear;
            until menucop = true;
          end;
        end;


      procedure addtetinfo;

begin
  clear;
  writeln(dots);
  lines(2);
  getfile('addtetinfo.text');
  lines(2);
  spacebar;
  clear;
  lines(2);
  getfile('addtinfl.text');
  lines(2);
end;



        procedure newtarget;

        var j, i : integer;

        begin
          recnum := 0;
          while tgtmap[recnum] <> 'zzzzz' do
          begin
            recnum := recnum + 1;
            if recnum = maxtgtat - 1 then
            begin
              clear;
              lines(11);
              writeln('        FILE FULL');
              lines(2);
              writeln('        Targets must be deleted in order to continue');
```

```
            lines(-);
            spacetar;
            done := true;
            exit(nextlevel);
          end;
        eri;
        seek(f1,recnum);
        get(f1);
        currentf1 := f1^.queuerec;
        seek(target,recnum);
        get(target);
        currentt := target^.textrec;
        for i := 1 to maxtrlevel do
          currentt.lines[i] := 0;
      eri;



      procedure nextlevel;

      var  i : integer;

begin
  maxtrlevel := 10;
  paramterwidth := 10;
  current := maxsm;
  outprocess := false;
  filsprot := false;
  done := false;
  quit := false;
  ok := true;
  clear;
  if first then
    begin
      writeln('write');
      lines(1);
      setfile('protest.text');
      prompt;
      first := false;
    end
  else
    begin
      lines(1);
writeln('  ** for help enter type a ?.....to continue, type return ');
      prompt;
    end;
  repeat
  read(ch);
  if ch = '?' then
      begin
       help;
       return;
```

```
      end;
 until eoln(input);
readln;
if eoln then writeln(interact);
while not eof do
begin
  for n := 1 to numbervar - 2 do
   begin
     if eof then writeln(interact);
     if interact.item[n] = on then
     begin
       case n of
       1 : tmthm;
       2 : tmtdc;
       3 : tmtdsc;
       4 : tmtclass;
       5 : tmtprl;
       6 : tmtstatus;
       7 : tmtmtype;
       8 : begin
             remainforwriter := false;
             tmtmit;
           end;
       9 : cmassert;
      10 : remarks;
      11 : mapper;
      12 : source;
      13 : erroron,
      14 : pnotcrit;
      15 : directive,
      16 : tmtaccuracy,
     end;{case}
   end;
   finished := false;
   end; {for}
   if fip <> 0 then checkfile
   else done := true;
  end; {while}
  process;
end;


      procedure tgtmend;

      begin
        clear;
        writeln(stars);
        lines(2);
        writeln('          available on the output file ');
        lines(1);
        writeln('      the options are:');
        lines(1);
        writeln('            1.  add a tmtmit');
```

```
        writeln('              2.  change a target');
        writeln('              3.  display a target');
        writeln('              4.  enter a target surveillance');
        writeln('              5.  delete a target');
        writeln('              6.  list targets currently in file');
        writeln('              7.',return);
        lines(2);
      end;


procedure listtargets;

var  j : integer;

    begin
      j := 2;
      clear;
      lines(2);
      writeln('                 list of current targets in file');
      lines(2);
      for i := 1 to numberat do
        if tgtrep[i] <> 'zero' then
          begin
            write('          ',tgtrep[i]);
            j := j + 1;
            if j = 5 then
              begin
                writeln;
                j := 2;
              end;
          end;
      lines(2);
      writeln('                    End of target list');
      lines(1);
      spacebar;
    end;



begin {target?(s)}
  nostring := '      ';
  number := ' ';
  first := true;
  mandatoryitem := false;
  menuloop := false;
  menuset := ['1','2','3','4','5','6','7','8','9','A','B','C','D','E','F','G'];
      repeat
       curr nl := false;
       tgttemp;
       select;
       read(ch);
       if ch in menuset then
       begin


```

```
      if eof(input) then readln;
      case ch of
        'i','a','s' : addtoeffort;
        'n','o','c' : checkset;
        'd','l',' ' : display;
        'g' : begin
                randomvalues := false;
                newsdn;
              end;
        'b' : deleteset;
        'c' : listcheck;
        'q','x','f' : exit(targetproc);
        '?' : begin
                clear;
                lines(5);
                settfile('methodx.text');
                lines(5);
                spaceback;
              end
      end
    else readerror;
  until (ch=blank) = true;
end;
```

```
                (*    INITIALIZE    *)

segment procedure initialize;

    var
          menchr : mvalue;
          goback : boolean;

     procedure initinfo;

     begin
clear;
writeln(stars);
lines(1);
writeln('   If the system has not been initialized when the program is');
writeln('started, it will initialize automatically and create the target');
writeln('files on the diskette. This procedure allows you to re-initialize');
writeln('the system for a new operation or to restart target information');
writeln('operations from a fresh start.');
lines(2);
writeln('   Option 2 will delete all the current files and allow you to');
writeln('start out from the beginning. The files you delete are not');
writeln('recoverable. Be sure you want to delete all of your files before');
writeln('you select option 2. Use option 3 to return to the main');
writeln('command menu. Pressing the return key will return you to the');
writeln('initialize option menu.');
lines(2);
returnbar;
end;


procedure initfile;

     var  i, j : integer;

begin {initfile}
  restart := true;
  write(dot);
  j := 1;
  write(dot);
  close(target,purge);
  rewrite(target,'#5:targetfile.data');
  for i := 1 to numbertgt do
    begin
      j := j + 1;
      if j = 15 then
      begin
        write(dot);
        j := 1;
      end;
      target^.tgtrec := emptyrec;
      put(target);
    end;
```

```
  close(target,lock);
  reset(target,'#5:targetfile.data');
  write(dot);
  close(QT,purge);
  rewrite(QT,'#5:queryfile.data');
  write(dot);
  for i := 1 to numbertgt do
    begin
      j := j + 1;
      if j = 15 then
      begin
        write(dot);
        j := 1;
      end;
      QT^.querrec := emptyQrec;
      put(QT);
    end;
  close(QT,lock);
  reset(QT,'#5:queryfile.data');
  write(dot);
end;


  procedure reinitialize;

  begin {reinit}
    goback := false;
    clear;
    lines(4);
writeln('        THIS PROCEDURE WILL DELETE ALL TARGET FILES.');
writeln(dots);
lines(1);
writeln(chr(7));
writeln('          The options are:');
lines(1);
writeln('                1.  Return to Main Command Menu');
lines(1);
writeln('                2.  Reinitialize System');
lines(1);
writeln('                3.  Information');
lines(1);
    select;
    repeat
    read(ch);
    if eoln (input) then
    begin
      goback := true;
      exit(reinitialize);
    end;
    if ch in ['1'..'3'] then
    begin
      case ch of
      '1' : exit(reinitialize);
      '2' : begin
```

```
            lines(2);
            write('        System initializing....');
            [illegible];
            initfile;
            lines(2);
            exit(reinitialize);
          end;
      'X','?' : initinfo;
        end
    end
    else [illegible];
    until gotans = true;
  end;{reinit}


          begin {initialize}
            gotans := false;
            answer := ['1','2',[illegible],'?'];
            if fileexists then
              begin
                initfile;
                exit(initialize);
              end;
          repeat
            [illegible];
            [illegible];
            lines(2);
            writeln('               Initialize the System');
            lines(2);
            writeln('     The [illegible]');
            lines(1);
            writeln('          1. [illegible]');
            lines(1);
            writeln('          2. [illegible]');
            lines(1);
            writeln('          [illegible]');
            [illegible];
            [illegible];
            [illegible];
            if [illegible] then
            begin
              [illegible]
              [illegible]
              '1','?' : initinfo;
              '2' : begin
                        initialize;
                        [illegible]
                    end;
              [illegible] : [illegible];
              end
            end
          else [illegible];
          until [illegible];
        end;
```

```
student procedure inform;

var
letchar : char;


  procedure informmain;

  begin
  clear;
  writeln(dots);
  lines(2);
  writeln('  this section provides information about the following:');
  lines(2);
  writeln('          1.  how to operate the system');
  writeln('          2.  Security requirements');
  writeln('          3.  Target Classifications');
  writeln('          4.  Target priorities');
  writeln('          5.  Target Analysis procedures');
  writeln('          6.',return);
  lines(2);
  end;


  procedure usertest;

  begin
    clear;
    writeln('          System operation Instructions');
    writeln(dots);
    lines(12);
    writeln('  TO GO BACK');
    lines(5);
    spacebar;
  end;


  procedure format1; forward;
  procedure formatoptions; forward;

  procedure formats;

  begin
    clear;
    lines(1);
    writeln('          formats used in the displays');
    writeln(dots);
    lines(1);
writeln('    There are two basic formats used in the target information ');
```

```
writeln('system. The first is a replica of the original target list as');
writeln('specified in step 2-1 (site request correlation . All of the');
writeln('information about a particular target, including target');
writeln('surveillance, can be displayed on the screen or printed on the');
writeln('line printer.');
lines(4);
writeln('     The second type of display is the target listing. This');
writeln('listing contains the most important items from the target');
writeln('data set is used primarily by the surveillance aircraft of');
writeln('the FSCC. This listing is available in two different forms.');
writeln('from the target list to a special listing of identified target');
writeln('characteristics.');
      lines(2);
      statusr;
      format1;
    end;


    procedure format1;

    begin
    clear;
    lines(1);
    writeln('     The procedure in this system allows you to display or');
    writeln('print a list of targets with a combination of the following');
    writeln('parameters: status, priority, classification, specific');
    writeln('air assigned, target type and area. For example, you');
    writeln('could obtain a list of all hostile, air target assigned');
    writeln('to naval airline for the naval airline attack. All the mis');
    writeln('is and ill identified targets assigned to airline for the');
    writeln('airline, either of all a list and targets for the air');
    writeln('airline.');
    lines(2);
    writeln('     A third item that is the display of printer is the');
    writeln('target tracking charts. The system maintains a');
    writeln('track of all transactions for the target list and allows');
    writeln('you to print a formatted tracking chart for transmission.');
    writeln('It uses the standard format of targets added to the list,');
    writeln('deleted from the list,targets changed and important target');
    writeln('surveillances.');
    lines(1);
    sbaccter;
    formatbtions;
    end;


    procedure fcardis; forward;
    procedure flistdis; forward;
    procedure ftrkdis; forward;
    procedure formcomm; forward;


    procedure formatbtions;
```

```
begin
repeat
  formopment;
  select;
  read(cc);
  if cc in [menchar] then
  begin
    if eoln (input) then readln;
    case cc of
      '1' : [illegible];
      '2' : [illegible];
      '3' : [illegible];
      'e','E','r' : exitformoptions ;
      'n','N' : [illegible];
      '?' : begin
            lines(2);
writeln('  The three options here will display the [illegible] and [illegible]');
writeln('formats for the target card, the target list and the target ',
writeln('bulletin. Choose one of these three options or type ' list ',',
writeln('to continue operations and return to the previous menu.');
            lin s(2);
            returnter;
            end
    end
    else menuerror;
until [illegible] = true;
end;



procedure formopment;

begin
  clear;
  writeln(scts);
  lines(2);
  writeln('              The display options are:');
  lines(2);
  writeln('                  1.  Target Card');
  writeln('                  2.  Target Listing');
  writeln('                  3.  Target Bulletin [illegible]');
  writeln('                  4.',return);
  lines(2)
  end;


  procedure Tcardadd;

  begin
writeln('Photo No:                    Map Ref   : IRAN 4577-IV');
writeln('Photo Coord:                 Accuracy  : CONFIRMED');
```

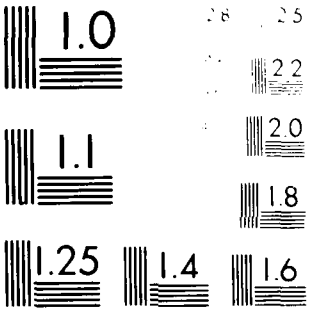
```
writeln('Remarks: First tank sighting in sector IV. Attack w/ rocket');
lines(1);
writeln('-------------------STRIKEBANCE-------------------------');
writeln('              Firing      NO/type        Damage       Damage');
writeln('   LTG        Unit        Rounds         Reported     Assessed');
writeln('   ---        ------      -------        --------     --------');
writeln(' 1216622    2 F/A-18     2  D-22        DESTROYED    DESTROYED');
writeln('FDA: Both tanks confirmed by AC. No AAA coverage on tgt.');
        lines(1);
        end;


     procedure lcardls;

     begin
        clear;
writeln(dots);
writeln('                    : TARGET NO.  AD0012 :');
writeln('                    -------------------- -');
writeln('Location: 34567875          Alt: 0K     Type: TANK');
writeln('Description: 2 T-62 TANKS IN OPEN FIELD');
writeln('Class    : A                          Status : ACTIVE');
writeln('Priority : 1                        Tgt List: YES');
writeln('SAassgn  : AIR');
lines(1);
writeln('Source of Tgt: AIR OBSERVER OV-10');
writeln('LTG Activated: 1216622');
        lcardall;
        spacebar;
        end;


     procedure llistals;

     begin
        clear;
writeln(stars);
lines(2);
writeln('                TARGET LISTING');
lines(1);
writeln('TGT NO  CL  PRI LOCATION  ALT   SAASG    DESCRIPTION');
writeln('------  --  --- --------  ----  ------   -----------');
writeln('AD0021* A    11 34556677   9K    AIR      2 T-62 TANKS IN OPEN');
writeln('AD0024* A     1 34524655   12K   NGF      FORTIFIED BUNKER COMPLEX');
writeln('AD0033* B    11 34555654   45    ARTY     BTY OF T-12K AT GUNS');
writeln('AD0054  F    10 34555566   1K    NGF      BRACE FORTIFICATIONS');
writeln('AD0055* A    10 34455770   5K    NONE     SCHOOL BUILDINGS');
writeln('NA0211  C   111 43225555   0     NGF      BRACE FORTIFICATIONS');
writeln('AD0057* A     1 34775545   12K   AIR      BTY 2S1-M6-4 IN TRENCHES');
writeln('AD0057* B    10 35557555   145   NONE     RAIL/SUPPLY DEPOT');
writeln('AD0056  F   111 3444e9KK   2K    ARTY     BTY LUG IN TRENCHLINES');
writeln('AD0058* A     1 34225596   7K    AIR      12 VEHICLES ALONG ROAD');
        lines(2);
```

```
        writeln('   NOTE: * indicates target list');
        lines(1);
        spacebar;
      end;


      procedure ?nuicisi;

      begin
clear;
lines(2);
writeln('    3.  CANCELLED TARGETS');
writeln('        AD2084, AD2750, AD2850, AD2122, AA2021');
writeln('        AD2046, AD2047, AD2107');
lines(1);
writeln('    4.  DEACTIVATED TARGETS');
writeln('        AD2677, AD2103');
lines(1);
writeln('    5.  CLASSIFICATION/PRIORITY CHANGES');
writeln('         AD2053   A   II');
writeln('         AD2054   C   IV');
writeln('         AD2079   D   III');
writeln('         AD2107   B   IV');
writeln('         AD2121   E    I');
writeln('         AD2221   A   II');
lines(1);
writeln('                                                    CLASSIFICATION');
lines(2);
spacebar;
      end;


      procedure ?nuicis;

      begin
      clear;
writeln('                    TARGET BULLETIN');
lines(1);
writeln('CLASSIFICATION');
lines(1);
writeln('    DTG   :  121052Z');
writeln('    FROM  :  CFL  (CTF32.1.1)');
writeln('    TO    :  DISTRIBUTION');
lines(1);
writeln('    SUBJ  :  TARGET NUMBER 12');
lines(1);
writeln('    1.  NEW TARGETS');
writeln('        AD2134  04555544  FORTIFICATIONS     A  I');
writeln('        AD2135  04526577  2 T-34 TANKS       A  I');
writeln('        AD2136  04507062  BUNKER COMPLEX     I  III');
lines(1);
writeln('    2.  BDA');
writeln('        AD2078  BZ2 damaged by air strike');
writeln('        AD2135  Destroyed');
```

AD-A112 429 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 15/4
A PROTOTYPE PROGRAM FOR TARGET INFORMATION.(U)
JUN 81 R J COULTER
UNCLASSIFIED NPS52-81-007 NL

2 OF 2
AD A
112429

END
DATE
FILMED
04-82
DTIC


1.0
2.8
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

```
writeln('        ADWLIB  Partially damaged by artillery');
lines(1);
spacebar;
Tbuldisl;
end;



     procedure tgtinfo;

     begin
       clear;
       writeln('              Target listing information');
       writeln(dots);
       lines(1);
       getfile('queryinfo.text');
       lines(1);
       spacebar;
     end;

      procedure version1; forward;

      procedure versadd;

      begin
writeln('     CRT CONSOLE: Datamedia Elite 25WE');
writeln('        LANGUAGE: Pascal');
writeln(' IMPLEMENTATION: UCSD Pascal (version 1.4p)');
writeln('          DESIGN: LtCol R. J. Coulter, USMC');
writeln('     PROGRAMMING: LtCol R. J. Coulter, USMC');
lines(1);
spacebar;
      end;

      procedure version;

      begin
      clear;
      writeln(dots);
writeln('  MIcrocomputer System for Target Information  (MISTI)');
writeln('  ----------------------------------------------------');
writeln('                  Version 1.0');
lines(1);
writeln(' A prototype microcomputer data base operation system for the');
writeln('     target information section of the Marine Corps fire support');
writeln('     coordination center. It is the result of a masters thesis');
writeln('     submitted at the Naval Postgraduate School.');
lines(1);
writeln('        LOCATION: Department of Computer Science');
writeln('                  Naval Postgraduate School');
writeln('                  Monterey, California');
writeln('            DATE: 1 June 1981');
writeln('SOURCE COMPUTER: Altos ACS 8000-1');
writeln('OBJECT COMPUTER: Altos ACS 8000-1');
```

```
versall;
version1;
end;


procedure version1;

begin
clear;
lines(4);
writeln(' System supports upper and lower case. Character delete key');
writeln('      is <runout> key. Input terminator is <return> key.');
lines(1);
writeln(' FOR INFORMATION:');
lines(1);
writeln(' Professor Lyle A. Cox,Jr.');
writeln('            Naval Postgraduate School');
writeln('            Monterey, California 93940');
writeln('            408-646-2449');
lines(1);
writeln('      LtCol Ronald J. Coulter, USMC');
writeln('            Development Center');
writeln('            MCDEC');
writeln('            Quantico, Virginia 22134');
writeln('            PHONE');
lines(3);
spacebar;
end;

     procedure sysopmenu;

     begin
       clear;
       writeln(dots);
       lines(1);
       writeln('                System Operation');
       lines(1);
       writeln('      The options are:');
       lines(1);
       writeln('          1.  Instructions for the User');
       writeln('          2.  Formats Used in Displays');
       writeln('          3.  Obtaining Information about Targets');
       writeln('          4.  System Technical Information');
       writeln('          5.',return);
       lines(2);
     end;


     procedure systemop;

     begin
       repeat
         sysopmenu;
```

```
        select;
        read(ch);
        if ch in tenchar then
        begin
          if eoln (input) then readln;
          case ch of
            '1' : userinst;
            '2' : formats;
            '3' : tgtinfo;
            '4' : version;
            '5','P','p' : exit(systemop);
            '6' : menuerror;
            '7' : begin
                    lines(1);
writeln('    Five options concerning system operation are provided in');
writeln('in the above menu. Select the item you want from these options');
writeln('and type that number on the keyboard. If you do not desire any');
writeln('information on system operations, then use option 5 to return');
writeln('to the previous menu');
                    lines(1);
                    returnbar;
                    end
            end
          end
          else menuerror;
        until menuloop = true;
      end;



    procedure security;

    begin
    clear;
    writeln('            Security Guidelines');
    writeln(dots);
    lines(10);
    writeln('   TO BE IMPLEMENTED');
    lines(7);
    spacebar;
    end;


    procedure tgtclass;

    begin
      clear;
      lines(1);
      getfile('class.text');
      lines(1);
      spacebar;
    end;
```

```
procedure tetpri;

begin
  clear;
  lines(1);
  getfile('priority.text');
  lines(3);
  spacebar;
end;


procedure anal1; forward;
procedure anal2; forward;
procedure anal3; forward;
procedure anal4; forward;
procedure anal5; forward;

procedure tetanal;

begin
clear;
lines(1);
writeln('                    Target Analysis Guidelines');
writeln(dots);
lines(1);
writeln('  The following format ensures a logical and orderly examination');
writeln('of all factors to determine the best method of attack of a target.');
lines(1);
writeln('Situation of opposing forces:');
writeln('----------------------------');
writeln('Enemy situation...include information that will aid target analysis.');
lines(1);
writeln('Friendly situation...information that will aid attack of the target.');
lines(1);
writeln('Target characteristics:');
writeln('-----------------------');
writeln('Target description....type(personnel, materiel, terrain), number');
writeln('                    of personnel, quantity of materiel and activity.');
lines(1);
writeln('Vulnerability...type and amount of cover, type of materiel, type');
writeln('               of construction, mobility and density of personnel');
writeln('               and material.');
lines(1);
spacebar;
anal1;
end;

procedure anal1;

begin
clear;
lines(1);
writeln('Physical location....grid reference, altitude of target, location');
```

```
writeln('                    of friendly forces and terrain features.');
lines(1);
writeln('Accuracy....of the target location and the agency reporting the');
writeln('              target.');
lines(1);
writeln('Size of area...dimensions and shape of the target area and the');
writeln('              distribution of personnel and material in the area.');
lines(1);
writeln('Terrain and weather...brief analysis of terrain and weather');
writeln('               in the target area. Include any terrain features');
writeln('               which affect the means and method of attack.');
lines(2);
writeln('Target Capabilities:');
writeln('--------------------');
writeln('    The capabilities of the target as they affect the accomplishment');
writeln('  of the mission of the supported unit. Show how a terrain feature');
writeln('  affects enemy capabilities.');
lines(1);
spacebar;
anal2;
end;

procedure anal2;

begin
clear;
lines(2);
writeln('Other Factors:');
writeln('--------------');
writeln('  How do the following affect the  action of firepower, method');
writeln('of attack and delivery means?');
lines(1);
writeln('Urgency of attack....determined by the type of target (static or');
writeln('              fleeting) and its capabilities.');
lines(1);
writeln('Enemy counterpressures...ability of the  army to minimize the');
writeln('              effects of firepower, prevent delivery of weapons');
writeln('              arms and bring counterpressures against military  ');
writeln('              after attack.');
lines(1);
writeln('Enemy discipline...factors which will aid in determining the');
writeln('              amount of firepower required to neutralize the target');
writeln('              and discipline of enemy personnel.');
lines(1);
writeln('Obstacles....considerations concerning the possibility of');
writeln('              creating obstacles by attacking the target.');
lines(1);
spacebar;
anal3;
end;


procedure anal3;
```

```
begin
clear;
lines(2);
writeln('Civilian casualties...approximate number of civilians in the');
writeln('        target area and the estimated amount of civilian casualties.');
writeln('        Casualties.');
lines(1);
writeln('Surprise....methods designed to obtain surprise in order to');
writeln('            expedite time of attack, reduce the [illegible]');
writeln('            restrictions of antiballistic [illegible].');
lines(1);
writeln('Means of Attack:');
writeln('----------------');
writeln('     All available types of [illegible] that can be used to [illegible]');
writeln('with which it is practical to attack the target [illegible]');
writeln('cost effectiveness [illegible].');
lines(1);
writeln('Analysis of Means of Attack:');
writeln('----------------------------');
writeln('  The effect of each [illegible] that is [illegible] [illegible]');
writeln('target capabilities [illegible] [illegible].');
writeln('Estimate the results of the [illegible].');
lines(1);
spacebar;
enable;
end;

procedure [illegible];

begin
clear;
lines(2);
writeln('1.  Location of [illegible] [illegible] [illegible].');
lines(1);
writeln('2.  Effect of available [illegible].');
lines(1);
writeln('3.  Estimate of [illegible] casualties [illegible].');
lines(1);
writeln('4.  Estimate of civilian casualties.');
lines(1);
writeln('5.  Estimate of obstacles [illegible].');
lines(1);
writeln('6.  Precautions required for [illegible].');
lines(1);
lines(1);
writeln('Comparison of Means of Attack:');
writeln('-----------------------------');
writeln('   The outstanding alternatives are displayed [illegible]');
writeln('of attack and determine which offers the greatest [illegible]');
writeln('success.');
lines(1);
spacebar;
enable;
end;
```

```
procedure [illegible];

begin
clear;
lines(2);
writeln('[illegible]:');
writeln('---------------------------------');
lines(1);
writeln('1.  [illegible].');
lines(1);
writeln('2.  [illegible].');
lines(1);
writeln('3.  [illegible].');
lines(1);
writeln('4.  [illegible].');
lines(1);
writeln('5.  [illegible] procedures, [illegible].');
lines(1);
writeln('6.  [illegible].');
lines(4);
returnkey;
end;




    begin {menu}
       menuchar := ['1','2','3','4','5','6','7','8','9'];
     repeat
       intermenu;
       select;
       read(ch);
       if ch in menuchar then
          begin
          if eoln(input) then readln;
          case ch of
              '1' : system[illegible];
              '2' : [illegible];
              '3' : [illegible]class;
              '4' : [illegible];
              '5' : [illegible];
              '6','7','8' : exit (menu);
              '9' : begin
lines(1);
writeln('  Six options are provided in the menu. [illegible]');
writeln('the item you want from these options. Type [illegible]');
writeln('on the keyboard and the system will respond with the');
writeln('desired information. If you do not [illegible]');
writeln('then type 9, [illegible].');
lines(1);
returnkey;
```

```
(*            UTILITY.TEXT            *)

segment procedure utility;

    var
          temchar : myset;
          t : integer;

 procedure checkcr;


procedure rainstl;

  begin
  lines(1);
writeln('     The password is changed by deleting the old password, substituting ');
writeln('an blank and writing the new password to the disk file. At the ');
writeln('end of an operation it is suggested that the password be changed.');
writeln('This will return them to their original user status. Initially ');
writeln('the system will automatically this.');
   lines(1);
   end;


 procedure initst;

     begin
       clear;
       lines(6);
writeln('     There are 2 passwords in the talent directory.');
writeln('One is the system password which cannot be changed. This will');
writeln('ensure that at least one of the passwords will always ');
writeln('  be the same as the system password, which is the ');
writeln('passwords can be input in the file to allow the ');
writeln('to have exclusive access to the target file.');
lines(1);
writeln('     Initially, these 2 user passwords are blank and remain that ');
writeln('until changed by this procedure. The passwords can be ');
writeln('to 12 letters or numbers. Examples of passwords could be the ');
writeln('personal last names (JOHN, JANET, FAITH), special words ');
writeln('numbers (.007, 1984) or any combination of ');
       rainstl;
       spacebar;
     end;


procedure currentPW;

begin
  clear;
  lines(4);
  writeln('          CURRENT PASSWORDS');
```

```
  writeln('          ------------------');
  lines(1);
  writeln('          1.   ',userid[1]);
  writeln('          2.   ',userid[2]);
  writeln('          3.   ',userid[3]);
  writeln('          4.   ',userid[4]);
end;


  procedure pwchange;

  var  item : integer;
       ok : boolean;
       newpw : string[12];

  begin
    ok := false;
    currentpw;
    writeln('          0.',return);
    lines(2);
    writeln('      ENTER THE NUMBER OF THE PARM YOU WISH TO CHANGE');
    prompt;
    read(item);
    if item in [1..4] then
    begin
      if eoln(input) then readln;
      if item = 0 then exit(pwchange);
      repeat
        lines(1);
        writeln('      ENTER NEW PASSWORD.....[illegible]');
        prompt;
        readln(newpw);
        if length(newpw) > 1. then
        begin

        {   function   }

          lines(1);
 writeln('     The password can be up to 12 letters or numbers in length,');
 writeln('such as a name, SSN or letter-number combination. Other than ');
 writeln('characters in the space after the system prompt and press the ');
 writeln('RETURN key. The prior password will be automatically replaced by the');
 writeln('new password substituted for it. If you just press the RETURN key ');
 writeln('the prior password will remain.');
          lines(1);
        end
        else ok := true;
        until ok = true;
        end
        else if length(newpw) = 0 then exit(pwchange);
        lines(2);
        writeln('       Password Changed');
        lines(2);
```

```
    return(r);
  end;


begin {main program}
  repeat
    clear;
    lines(0);
    writeln('                    Changing the [illegible]');
    writeln('[illegible]');
    lines(1);
    writeln('          The options are:');
    lines(1);
    writeln('            1.  Instruction [illegible]');
    writeln('            2.  Display a current [illegible]');
    writeln('            3.  Change password');
    writeln('            4. ,return');
    lines(1);
    select;
    read(c);
    if c in [1..4] then
    begin
      if c=chr(13) then read(c);
      case c of
        '1','2' : [illegible];
        '3' : begin
                [illegible];
                lines(2);
                [illegible];
              end;
        '4' : [illegible];
        '4', '5', '6', '7' : writeln('[illegible]');
        '8', '9', '0', '-', '=' : [illegible];
      end;
    end
    else [illegible];
  until [illegible] = true;
end;


procedure [illegible];

      var  [illegible] : integer;


procedure display[illegible];

  begin
    clear;
```

```
  lines(1);
  writln('                    [illegible]');
  for i := 1 to 10 do
  begin
    writln(i);
    lines(1);
  end;
  clear;
  writln('                    [illegible]');
  lines(1);
  writln('THE SIMPLEST [illegible]');
  lines(1);
  writln('    [illegible] :');
  writln('    [illegible] :');
  writln('    [illegible] : [illegible]');
  writln('    [illegible] [illegible]');
  lines(1);
  writln('    1. [illegible]');
  writln('                [illegible]');
  writln('    [illegible]');
  writln('                [illegible]');
  writln('    [illegible]');
  writln('                [illegible]');
  writln('    [illegible] [illegible]');
  writln('                [illegible]');
  writln('    [illegible] [illegible]');
  writln('                [illegible]');
  writln('    [illegible] [illegible]');
  writln('                [illegible]');
  lines(1);
  writln('                                    [illegible]');
  lines(1);
  prompt;
end;


procedure [illegible];

var  i, amount : integer;

begin
  amount := 0;
  clear;
  lines(2);
  writln('  [illegible]');
  writln('              [illegible]');
  lines(2);
  writln('  Please enter the [illegible]');
  writln('              [illegible]');
  lines(1);
  prompt;
  readln(amount);
  if amount <> [illegible]
  lines(2);
  [illegible]
```

```
  writeln('[illegible]');
  lines(2);
  for [illegible]
    [illegible]
      lines;
      writeln('[illegible]');
      writeln('[illegible]');
    [illegible];
  writeln('[illegible]');
  writeln('[illegible]');
  lines(1);
  writeln('[illegible]');
  writeln('[illegible]');
  lines(2);
  spacebar;
end;


procedure [illegible];


begin
  [illegible];
  [illegible];
  lines(2);
  writeln('[illegible]');
  lines(1);
writeln('[illegible]');
writeln('[illegible]');
writeln('[illegible]');
writeln('[illegible]');
lines(2);
writeln('[illegible]');
  prompt;
  readln([illegible]);
  if [illegible] = 'y' then [illegible];

  [illegible] process [illegible]

  lines(4);
  writeln('[illegible]');
  lines(1);
  writeln('[illegible]');
  lines(1);
  spacebar;
end;


procedure [illegible];

begin

```

```
  repeat
    read;
    writln('        Start - [illegible] TAPE [illegible]');
    writln(lists);
    lines(1);
    writln('        The options are:');
    lines(1);
    writln('          1. [illegible] a new TAPE file');
    writln('          2. Interpret');
    writln('          3.', [illegible]);
    lines(1);
    select;
    read(k);
    if ch in ['1','2','3','4','5','6'] then
    begin
      if eoln(input) then readln;
      case ch of
        '1': begin
               lines(1);
               writln('        Creating a new TAPE file.....');
               lines(2);
               [illegible] if [illegible] = 1 ;

               [illegible]

               writln('    ** [illegible] ');
               lines(2);
               [illegible]
               [illegible]
             end;
        '2','3' : begin
                    lines(1);
writln('    After the TAPE is [illegible] [illegible] [illegible]');
writln('[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]');
writln('[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]');
writln('a new TAPE file [illegible] [illegible] [illegible] [illegible] [illegible]');
                    lines(1);
                    [illegible];
                  end;
        '4','5','6' : [illegible]
      end
    end
    else [illegible];
  until [illegible] = true;
end;


procedure Initp;

  begin
    lines(4);
writln('    The transitions of the [illegible] [illegible] [illegible]');
writln('over [illegible] in the TAPE file. [illegible] [illegible]');
writln('the FSM, the [illegible] [illegible] [illegible] [illegible]');
```

```
writeln('can be written into this file. It contains [illegible]');
writeln('and changes to the latest list since the [illegible]');
lines(1);
writeln('     After the [illegible] is written, you first [illegible] a [illegible] the');
writeln('the system will [illegible] you at that [illegible] the [illegible] [illegible]');
writeln('you set the [illegible] after the first time, it [illegible] [illegible]');
writeln('the [illegible] number for each [illegible] [illegible].');
lines(1);
writeln('     Option 1 can be used to [illegible] [illegible] [illegible]');
writeln('[illegible]. Option 2 is used [illegible] [illegible] the [illegible]');
writeln('issued by the [illegible]. [illegible] [illegible] [illegible] [illegible].');
    lines(2);
  end;


begin {[illegible]}
  repeat
    clrscr;
    lines(1);
    writeln('                    [illegible] UTILITY');
    writeln('[illegible]');
    lines(1);
    writeln('        The options are :');
    lines(1);
    writeln('              1. [illegible] [illegible] [illegible]');
    writeln('              2. [illegible] [illegible] [illegible]');
    writeln('              3. [illegible] [illegible] [illegible]');
    writeln('              4. [illegible] [illegible] [illegible]');
    writeln('              5. [illegible]');
    writeln('              6. [illegible]');
    lines(2);
    select;
    read(ch);
    if ch in ['1'..'6'] then
    begin
      if ch in ['1'..'4'] then begin
      case ch of
        '1' : display[1];
        '2' : [illegible];
        '3' : [illegible];
        '4' : [illegible];
        '5','6' : begin
                    [illegible];
                    [illegible];
                    [illegible];
                  end;
        'q','Q','r' : exit([illegible]);
        '?','-' : [illegible];
      end
    end
    else [illegible];
  until [illegible] = true;
end;
```

```
procedure printlist;


procedure printmenu;

begin
  clear;
  writeln('                    MAIN TARGET LIST');
  writeln(chr(
  lines(1);
  writeln('     The options are:',
  lines(1);
  writeln('                  1.  Print the target list');
  writeln('                  2.  Print the list of targets');
  writeln('                  3.  Print the target [illegible]');
  writeln('                  4.  Print a target (s-1)');
  writeln('                  5.  Information');
  writeln('                  6.  Information on specific target list');
  writeln('                  7.',chr(
  lines(1);
  select;
end;


procedure printinfo;

  begin
    lines(1);
writeln('     The printing list prints all the [illegible]');
writeln('target in the target [illegible] to [illegible] on [illegible]');
writeln('to the system in case of a failure. If possible, all targets');
writeln('from the data base [illegible] in the printed in this [illegible]');
writeln('on printing lists with special parameters [illegible]');
writeln('the original [illegible] of the main command file.');
    lines(1);
  end;


procedure printlist;

begin
  clear;
writeln('     This procedure prints the formatted target [illegible]');
writeln('to the line printer for [illegible] purposes [illegible]');
writeln('for [illegible] personnel. The [illegible] format for the list is as [illegible]');
lines(1);
writeln('TGT NO  CI PRI LOCATION  ALT  SPEED  [illegible]');
writeln('------  - --- --------  ----  -----  [illegible]');
writeln('AB1230  A   1 [illegible]   [illegible]  [illegible]  [illegible]');
writeln('AB1240  [illegible]  19 [illegible]  1[illegible]  [illegible]  [illegible]');
writeln('AB1250  B  11 [illegible]   [illegible]  [illegible]  [illegible]');
writeln('AB2110  B   1 [illegible]   [illegible]  [illegible]  [illegible]');
```

```
writeln('Please [illegible]');
writeln('Alert [illegible]');
lines(1);
writeln('[illegible] = [illegible] [illegible]');
  printhead;
  specchar;
end;


procedure [illegible];

begin
 clear;
 lines(3);
 writeln('[illegible]');
 writeln('[illegible]');
 lines(2);
 writeln('[illegible]');
 writeln('[illegible]');
 writeln('[illegible]');
 lines(1);
 writeln('          [illegible]');
 writeln('          [illegible]');
 writeln('          [illegible]');
 writeln('          [illegible]');
 writeln('          [illegible]');
 writeln('          [illegible]');
 lines(1);
 writeln('[illegible]');
 writeln('[illegible]');
 writeln('[illegible]');
 writeln('[illegible]');
 lines(1);
 specchar;
 end;


procedure print.1;

begin
  clear;
  lines(4);
  write('        Printing the Current Files....');
  for t := 1 to 10 do
  begin
    delay;
    write(tot);
  end;
  lines(2);
  writeln('        ** Function Complete **');
  lines(1);
```

```
  spacebar;
end;


procedure printlist;

begin
  clear;
  lines(4);
  write('        Printing the list of [illegible]...');
  for t := 1 to [illegible] 10
  begin
    delay;
    write(dot);
  end;
  lines(8);
  writeln('          ** [illegible] **');
  lines(1);
  spacebar;
end;



procedure printall;

begin
  clear;
  lines(4);
  write('   Printing all active [illegible] ...');
  for t := 1 to [illegible] 10
  begin
    delay;
    write(dot);
  end;
  lines(8);
  writeln('          ** [illegible] **');
  lines(1);
  spacebar;
end;


procedure writeout;

var  tmpr : stringlol;

begin
  repeat
    outout := 'yes';
    clear;
    lines(2);
    writeln('      [illegible] [illegible]');
    prompt;
    readln(tmpr);
    if (tmpr = 'all') or (tmpr = 'ALL') then [illegible]
```

```
    else if length(time) < 8 then exit(printtest);
    else if length(time) < 10 then
    begin
      getout := true;
      lines(1);
writeln(' The target number consists of two letters or four digits.');
writeln(' For example, AZ or 1234. Please reenter data.');
lines(1);
writeln(' To print all of the target data, type ALL or [illegible]');
      lines(1);
      returnin;
    end;
  until getout = false;
  lines(0);
  write(' Printing the target data for target ',[illegible],'...');
  for t := 1 to 10 do
  begin
    delay,
    write(dot);
  end;
  lines(0);
  writeln([illegible]);
  lines(1);
  spacebar;
end;



begin {printtest}
  repeat
    printmenu;
    read(ch);
    if eoln(input) then readln;
    begin
      if eoln(input) then readln;
      case ch of
        '1' : print1;
        '2' : print2;
        '3' : print3;
        '4' : printall;
        'r','R' : printtrc;
        'o' : specialinfo;
        'q','Q','e' : exit(printtest);
        'z' : zeroerror;
      end
    end
    else zeroerror;
  until zeroerror = true;
end;


procedure stats;
```

```
begin
  clear;
  lines(4);
  writln('  processing Target Statistical Information.....');
  for t := 1 to 1[illegible] do
  begin
    delay;
    write(dot);
  end;
  lines(1);
  writln('        [illegible] Statistical Information [illegible] ');
  lines(4);
  returnch;
end;


procedure -this-info;

  begin
    lines(1);
writln('     This procedure will erase every file in the [illegible] ');
writln('all [illegible] the target information. [illegible] ');
writln('the erase directory. This is done [illegible] ');
writln('the classified information [illegible] the diskette.');
lines(2);
writln(' The initialization procedure will [illegible] ');
writln('declassifies the diskette and should only be [illegible] ');
writln('of an operation and the data is no longer [illegible].');
    lines(1);
  end;


procedure erase;

begin
  repeat
    clear;
    lines(1);
    writln('          Erasing the Target File');
    writln(dots);
    lines(1);
    writln('          The options are:');
    lines(1);
    writln('            1.  Introduction');
    writln('            2.  Erase file');
    writln('            3.',return);
    lines(1);
    select;
    read(ch);
    if ch in ['1','2','3','9','1','7'] then
    begin
      if eoln(input) then readln;
      case ch of
```

```
        '1','2' : begin
                    clear;
                    eraseline;
                    spacebar;
                  end;
        '2' : begin
                lines(6);
                write('      Erasing All Files.....');
                fileCheck := true;
                initialize;
                clear;
                lines(2);
writeln('                    ** Function Complete **');
lines(6);
writeln('      Diskette file erased....Return to main menu ...');
lines(2);
writeln(' **  Part operation by selecting option 7 on the main menu ...');
writeln('                    ..........then restart system');
                lines(2);
                spacebar;
                retout := true;
                exit(erase);
              end;
        '3','4','5' : exit(erase);
      end
    end
    else tenuerror;
  until menuloop = true;
end;


procedure copyPrompt;

  begin
    lines(2);
writeln('   **  Place the current target diskette in drive B **',
writeln('                        (front side)');
lines(1);
writeln('   **  Place the back-up diskette in drive A **');
writeln('                        (front side)');
lines(1);
writeln('   **  Press the RETURN key. The system will not respond');
writeln('       Copy the target information from drive B to drive A. ');
lines(1);
writeln('   **  When FUNCTION COMPLETE appears, do the following:');
writeln('         * Remove the back-up diskette from drive A');
writeln('         * Remove the target diskette from drive B');
writeln('         * Place the system diskette back in drive A');
writeln('         * Place the target diskette back in drive B');
writeln('         * Press the RETURN key');
lines(2);
writeln('      PRESS RETURN TO COPY DATA TO B');
    prompt;
```

```
  end;


procedure copyis;

var copydir : string[4];

begin
  clear;
  lines(1);
  writeln('              Copy Data Base Procedure');
  writeln(dots);
  lines(2);
writeln('   This procedure allows you to make a backup copy of');
writeln('   the target diskette. It requires you to switch the');
writeln('   diskettes in the disk drives and use a preformatted');
writeln('   back-up diskette. If you do not desire to copy the');
writeln('   data base, then press the RETURN key to return to the');
writeln('   previous menu. The directions in this section must be');
writeln('   followed exactly.');
  lines(3);
  writeln('   **  Type COPY and press the RETURN key');
  prompt;
  readln(copydir);
  if (copydir = 'COPY') or (copydir = 'copy') then
  begin
    clear;
    copyinfo;
    repeat
      read(c);
    until eoln(input);
    clear;
    lines(2);
    write('        COPYING DATA BASE...');
    for t := 1 to 10 do
    begin
      write(dot);
      delay;
    end;
    lines(3);
    writeln('     **  FUNCTION COMPLETE **');
    lines(1);
    returnter;
  end
  else exit(copyis);
end;




  procedure utilmenu;
```

```
begin
  clear;
  lines(1);
  writeln('              THE SYSTEM UTILITY FUNCTIONS',
  writeln(cots);
  lines(1);
  writeln('        The options are:',);
  lines(1);
  writeln('            1.  Change the password',);
  writeln('            2.  Copy the data base file',);
  writeln('            3.  Construct the Tabrul',);
  writeln('            4.  Print the tele-text',);
  writeln('            5.  Display target file statistics',);
  writeln('            6.  Erase the target files',);
  writeln('            7.  Information on the system times',);
  writeln('            8.',return);
  lines(1);
  select;
end;


procedure utinfo;

  begin
writeln('    The fourth option prints the list of targets, the tele-text',
writeln('and target information in the target file format. Printing is done',
writeln('by special parameters (like all these m. results) is done',
writeln('by a different procedure. The statistics display shows a number',
writeln('breakdown of the categories of information in the list of targets.',
lines(1);
writeln('    Option 6 erases all the information from the diskette. This is',
writeln('done at the end of an operation to reclassify the diskette files.',
writeln('The last option returns you to the main command menu.');
  lines(1);
  end;


     procedure utilinfo;

    begin
      clear;
      lines(1);
writeln('    This section provides various housekeeping procedures for',
writeln('the IIO. The first option allows you to change the passwords',
writeln('for the system users. The second permits you to copy the target',
writeln('files from the target diskette to a second diskette to function',
writeln('as a backup file. The third option constructs a target table',
writeln('(TABRUL) from all the data base transactions since the last TABRUL',
writeln('was printed. The routine will let you view or print TABRUL',
writeln('information and print it in the proper format for it.');
      lines(1);
      utinfo;
```

```
    spacetar;
  end;


begin {utility}
  menunum := ['1','2','3','4','5','6','7','8','x','r','?'],
  clear;
  getfile('kettab.text',);
  spacetar;
  repeat
    utilmenu;
    read(ch);
    if ch in menunum then
    begin
      if eoln(input) then readln;
      case ch of
        '1' : changetar;
        '2' : copytar;
        '3' : taltar;
        '4' : printtet;
        '5' : stats;
        '6' : begin
                erase;
                if setout then
                begin
                  setout := false;
                  exit(utility);
                end;
              end;
        '7','8' : utilmenu;
        'x','r','?' : exit(utility);
      end
    end
    else menuerror;
  until menuloop = true;
end;
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

:::::::::::::::::::::: [illegible] :::::::::::::::::

This [illegible] will [illegible] for each target. [illegible] appropriate information. These [illegible]

1. Target [illegible]
2. Grid location
3. Description
4. Initials
5. Classification
6. Status
7. Target type

Other items like map reference [illegible] left blank by merely pressing the RETURN key.

:::::::::::::::::::::: FUNIT1.TXT ::::::::::::::::::::::::::

The firing unit is the designation of the supporting unit which undertakes the attack of the target. The input can not exceed [illegible] letters or characters. An example of firing unit would be, Air... [illegible] artillery... [illegible], [illegible]... [illegible]. Enter the appropriate unit and press the RETURN key.

:::::::::::::::::::::: [illegible].TXT ::::::::::::::::::::::::

Enter the [illegible] target and indicate the type of [illegible]. [illegible] the input to 16 characters. [illegible] the number and type [illegible] required. If [illegible] press the RETURN key. An example of [illegible] artillery... [illegible], NGF... [illegible].

::::::::::::::::::::::::: FUNIT2.TXT ::::::::::::::::::::::::

The length of the input exceeds six characters.
please reenter the firing unit in six characters
or less and press the RETURN key

:::::::::::::::::::::::: DAMAGE.TXT :::::::::::::::::::::::

The options are:

1. Damaged
2. Destroyed
3. Interdicted
4. Harassed
5. Neutralized
6. Illuminated
7. Unobserved

[illegible]

:::::::::::::::::::::::: [illegible] ::::::::::::::::::::::::

The length of the [illegible] exceeds the [illegible] limit.
Please re-enter the [illegible] field in [illegible]
of [illegible] and [illegible] the following.

:::::::::::::::::::::::: [illegible] ::::::::::::::::::::::::

Enter the reported [illegible] of the above [illegible]. This is [illegible] to the [illegible] of the [illegible] which will [illegible] the target.

:::::::::::::::::::::::: [illegible] ::::::::::::::::::::::::

Enter the estimate of the actual [illegible] of the target [illegible] of the reported [illegible] [illegible]. It does not necessarily have to [illegible] with the [illegible].

:::::::::::::::::::::::: [illegible] ::::::::::::::::::::::::

The input in this file is [illegible] as the target [illegible].
Processing cannot continue without [illegible] the [illegible]
file. Please re-enter the data.

For information, type a ? after the entry [illegible].

:::::::::::::::::::::::: [illegible] ::::::::::::::::::::::::

The target number consists of two letters and [illegible]. An example of a correct entry would be [illegible]. [illegible] letters for the target designation. Please re-enter the data.

:::::::::::::::::::::::: [illegible] ::::::::::::::::::::::::

The grid coordinates of the target should [illegible] and consist only of numbers. For example, [illegible]. A six number coordinate such as [illegible] them one by adding zeros to [illegible]. Please re-enter data.

:::::::::::::::::::::::: [illegible] ::::::::::::::::::::::::

The target description can be up to [illegible] characters [illegible] include both numbers and letters. If the description exceeds [illegible] characters, then all characters [illegible] will be [illegible]. Be sure to indicate the quantity and the type of target. For example, [illegible] ZSU 23-4 in open field.

[illegible]

[illegible]

[illegible]

1. A
2. B
3. C
4. D
5. E

[illegible]

[illegible]

[illegible]

1. I
2. II
3. III
4. IV

[illegible]

[illegible]

[illegible]

1. Data
2. Fuel target
3. Installation
4. Coastal battery
5. Observation post
6. Terrain
7. Vehicles
8. Fortifications
9. Miscellaneous

[illegible]

The target type is entered and from the target description, information will be used to group targets of the same or similar type together. If the target cannot be classified into the above categories, use miscellaneous (option 9).

[illegible]

The altitude is entered in meters and consists of up to four digits from zero to 9999. For example, 400. The decimal must be entered in meters. If the altitude is not known, then press the return key. Please enter date.

[illegible]

[illegible]

::::::::::::::::::::::: [illegible] :::::::::::::::::::::::

[illegible]

::::::::::::::::::::::: [illegible] :::::::::::::::::::::::

[illegible]

::::::::::::::::::::::: [illegible] :::::::::::::::::::::::

[illegible]

::::::::::::::::::::::: [illegible] :::::::::::::::::::::::

[illegible]

::::::::::::::::::::::: [illegible] :::::::::::::::::::::::

If the target was obtained from an aerial photo enter the photo number and press the [illegible] key. For example, A211234. If the target [illegible] then just press the return key.

::::::::::::::::::::::: PHOTOGRAPHY :::::::::::::::::::::::

** If the photo grid location is the same as the target location, enter an 8 and press the return key. Otherwise, enter the eight digit number and press the return key.

ENTER PHOTO COORDINATES

::::::::::::::::::::::: PHOTOGRAPHY :::::::::::::::::::::::

The coordinates of the photo target should be [illegible] and consist only of numbers. For example, [illegible]. A six number coordinate such as [illegible] can be changed to [illegible] digit one by adding zeros to give [illegible].

If the photo grid location is the same as the target location

[illegible], enter or [illegible] press the return key. [illegible] [illegible].

::::::::::::::::::::::: [illegible] :::::::::::::::::::::::::

The [illegible] represent the [illegible] of the target information. A target which is known to exist is a [illegible] target. A probable and a possible target [illegible] to exist. If the intelligence evaluation does not fall into one of these areas, then enter option 4 for [illegible]. If [illegible] supplied, the system will automatically enter [illegible] the target [illegible].

::::::::::::::::::::::: [illegible] :::::::::::::::::::::::::

This procedure can only be used on targets [illegible] that [illegible] and exist in the list of targets. [illegible] you [illegible] the existing target information [illegible] the [illegible] you will be change. The system will [illegible] the [illegible] information and ask if you desire to change.

A [illegible] response will cause the [illegible] to be [illegible]. [illegible] the system [illegible]. The target [illegible] which will not be changed [illegible] key. [illegible] the [illegible] of [illegible].

Changes to the [illegible] of additions to the [illegible] [illegible] are permitted by [illegible] [illegible] [illegible] other [illegible] to the [illegible].

::::::::::::::::::::::: [illegible] :::::::::::::::::::::::::

The first option will display the [illegible] checks each item of target information and allows you to [illegible] those areas desired. The third option [illegible] the [illegible] file and the last option returns you to the [illegible].

::::::::::::::::::::::: PRIORITY.LIST :::::::::::::::::::::::

Target priorities

.....

PRIORITY I....Targets capable of [illegible] the [illegible] of the risk of action to the [illegible] and its elements.

PRIORITY II...Targets capable of [illegible] with the risk of action to the [illegible] and its elements.

PRIORITY III..Targets capable of [illegible] with the risk of action to the [illegible] and its elements.

PRIORITY IV...Targets capable of [illegible]

[illegible]
[illegible]
and its [illegible].

::::::::::::::::::::: [illegible] :::::::::::::::::::::

Option 1 will return you to [illegible]
[illegible] so that you may enter the [illegible]. [illegible]
allows you to proceed to the [illegible]
the target in the file, [illegible]
to the data of [illegible]
you to select the [illegible] option [illegible]

::::::::::::::::::::: [illegible] :::::::::::::::::::::

[illegible]
the after you have [illegible]
the data by the [illegible]

This procedure [illegible] Until [illegible]
you will [illegible] the [illegible]
file, displaying the [illegible]
[illegible] of [illegible] the [illegible] [illegible].

[illegible]

[illegible]

[illegible]
system prompt [illegible]
of example of the [illegible]

Follow all the [illegible]

::::::::::::::::::::: [illegible] :::::::::::::::::::::

This is the [illegible]

Option 1...provides information on how to [illegible]
[illegible]
requirements, inputs used, [illegible]
and other [illegible].

Option 2...enables you to add a target to [illegible]
the target file, [illegible]
and display [illegible]
target on the screen.

Option 3...enables you to attach a target [illegible]
a parameter or a list of parameters. [illegible]
include classification, [illegible]
inactive/attached), and [illegible]
All information for a [illegible]
to displayed.

::::::::::::::::::::: [illegible] :::::::::::::::::::::::::::

Option [illegible] allows you to [illegible] copy the [illegible] data base to another [illegible], check the data base, print [illegible] the [illegible], display the plane of [illegible] as well as provide a [illegible] breakdown of the [illegible].

Option [illegible] initializes the [illegible] system for a new operation. All [illegible] will be [illegible] of [illegible] new information can be entered. The [illegible] files will be [illegible].

Option [illegible] provides this [illegible] about the [illegible]. The [illegible] information can be obtained by [illegible] option.

Option 7 [illegible] the operation of the system after [illegible] important information to the [illegible] files. The system will have to be [illegible] [illegible] file.

::::::::::::::::::::: [illegible] :::::::::::::::::::::::::::

If you need help at any time, type [illegible].

If you want to return to the previous menu [illegible] option number followed by the [illegible] key.

::::::::::::::::::::: [illegible] :::::::::::::::::::::::::::

These procedures operate on the [illegible] list of targets. Option 1 allows you to add a new target to the list of targets. Option 2 allows you to change any information about a target [illegible] in the target file. Option 3 displays all the [illegible] about a particular target on the screen in a target data format. The target can be found either by target number or grid coordinates.

Option 4 allows you to add a new target surveillance [illegible] to the target based on the results of [illegible] supporting arms. Option 5 deletes a target completely from the list of targets. The final option returns you to the main menu. This menu also allows you to use the letter [illegible] for [illegible] targets, [illegible] for displaying targets and [illegible] for [illegible] targets.

::::::::::::::::::::: [illegible] :::::::::::::::::::::::::::

Enter the date-time-group that the [illegible] was [illegible] by supporting arms. The first [illegible] digits of the [illegible]

(1..31) and the next [illegible] the [illegible]. The letter indicates the time zone. [illegible], [illegible], [illegible]. If the time zone would be written, [illegible] enter the date and press the return key. If the [illegible] of activation is not known, then just press the return key.

:::::::::::::::::::: [illegible] ::::::::::::::::::::::::

The Battle Damage Assessment (BDA) can be up to [illegible] characters long. This is a [illegible] narrative of the surveillance of the target based on the observer report. If there is no report of observation, then this section may be skipped. An example of a BDA is: [illegible] secondary explosions, 2 vehicles burning.

:::::::::::::::::::: [illegible] ::::::::::::::::::::::::

Each target has three spaces for recording the target surveillance as a result of attack by supporting arms. This procedure prompts you for each piece of information for the Battle Damage Assessment (BDA). If a fourth BDA is entered, it will write over the first BDA since that is the oldest.

Information on multiple surveillances can be included in the remarks section of the target report by using the update target procedure. Use the display option to view the current BDA information in the target file.

:::::::::::::::::::: [illegible] ::::::::::::::::::::::::

Enter the date-time-group that the target was activated or added to the list of targets. The first 2 digits are the day of the current month (1..31) and the next 4 are the time (0001..2359). The letter indicates the time zone. For example, 1004 [illegible] day in the [illegible] time zone would be written, 1004[illegible]. Enter the date and press the return key. If the DTG of activation is not known, then just press the return key.

:::::::::::::::::::: [illegible] ::::::::::::::::::::::::

Target Classification
.....

CLASS A....Targets that threaten ships, aircraft, transshipping and UDT operations.

CLASS B....Targets that threaten assault forces in the ship-to-shore movement and assault of the beach.

CLASS C.....targets that interfere or cross friendly force operations either tactically or effect the ability of the enemy to continue resistance.

CLASS D.....targets that will not be fired upon prior to attack.

CLASS E.....targets that must not be destroyed because of probable future use of it exploited from cross authorized by the commander.

.....

::::::::::::::::: DEFAULT.DAT :::::::::::::::::::::::::::

ENTER TARGET ACCURACY

The options are:

1. Confirmed
2. Probable
3. Possible
4. Unknown

::::::::::::::::: SHOT.DAT :::::::::::::::::::::::::::

ENTER DEFENDING ATK ASSIGNED TO TARGET

The options are:

1. ATK
2. WDF
3. MIR
4. MIR, ATK
5. MIR, WDF
6. ATK, WDF
7. MIR, ATK, WDF
8. OTHER
9. NONE

::::::::::::::::: COMMAND.DAT :::::::::::::::::::::::::::

The options are:

1. Display the target data
2. Page through the target data
3. Write the change to the file
4. Return to previous menu

::::::::::::::::: PARTGT.DAT :::::::::::::::::::::::::::

INSTRUCTIONS FOR ADDING A TARGET

.....

The system will ask for each item of information.

Enter the required information and press the RETURN key.

To leave this procedure at any time, type a w after the entry prompt

To skip a section, just press a RETURN key and the system will go
to the next item of information unless the information requested
is mandatory. In that case you must enter information.

To receive more information about this procedure, type a ?

** To continue, press the RETURN key **

::

DISTRIBUTION

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
4.	Professor Lyle A. Cox, Jr., Code 52Cl Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
5.	Commanding General Attn: Fire Support Coordinator 1st Marine Division, FMF Camp Pendleton, California 92055	1
6.	Commanding General Attn: Fire Support Coordinator 2nd Marine Division, FMF Camp Lejeune, North Carolina 28542	1
6.	Commanding General Attn: Fire Support Coordinator 3rd Marine Division, FMF FPO San Francisco 96602	1
7.	Commanding General Education Center Attn: Supporting Arms Branch MCDEC Quantico, Virginia 22134	1

8. Commanding General 1
Development Center
MCDEC
Quantico, Virginia 22134

9. Commanding General 1
Attn: Fire Support Section (TEC)
MCAGCTC
Twentynine Palms, California 92227

10. Marine Corps Representative 1
U. S. Army Artillery School
Fort Sill, Oklahoma 73503

11. TRADOC Research Element Monterey, Code TRSM 1
Naval Postgraduate School
Monterey, California 93940

12. Commanding Officer 1
MCTSSA
Attn: MIPASS Team
Camp Pendleton, California 92055

13. LtCol Ronald J. Coulter, USMC 1
6033 Burnside Landing Drive
Burke, Virginia 22015

LMED

-8